**Universitetet i Oslo**
**Institutt for informatikk**

# Klustring av mikromatrisedata: Estimering av antall klustre og identifikasjon av subtyper.

Masteroppgave

Espen Solberg

**30. oktober 2007**

# Sammendrag

Et sentralt problem innen medisinsk forskning er å gruppere pasienter med hensyn til diagnose, prognose eller behandling, ut fra prøveresultater. Hvis prøveresultatene for en pasient består av genekspresjonsmålinger fra et mikromatriseforsøk, kan slike resultater fra flere pasienter arrangeres i en tabell som har en kolonne for hver observasjon (pasient/individ) og en rad for hvert gen. Dette gir en matrise hvor antall rader typisk er mye større enn antall kolonner. Hver pasient er i dette tilfellet representert som et punkt i et $p$-dimensjonalt rom, der $p$ er antall gener brukt i eksperimentet. Det er ikke enkelt å visualisere en slik situasjon, og det er heller ikke enkelt å bestemme hvilke observasjoner som faller inn i samme gruppe.

Å bestemme antallet og sammensetningen av individer i slike grupper (ofte kalt klustre) kan definere hvilke individer som etter et gitt sammenlikningsmål kan betraktes som mest like. Det finnes ulike velkjente metoder (klustringsmetoder) for å se på sammensetningen av slike observasjoner, men det er allikvel vanskelig å kunne bestemme gode klustre.

Sørlie *et al.* (2003) er et studium av mikromatrisedata, hvor genekspresjoner er hentet fra tumorer hos pasienter med brystkreft. Resultatet av dette studiet er en identifikasjon av 5 undergrupper av pasienter med hvert sitt karakteristiske genekspresjonsmønster, basert på både klustring og biologiske vurderinger.

Jeg har i denne oppgaven valgt å ta utgangspunkt i dette studiet, for å se på muligheten til å automatisere en slik klustring. En løsning på dette er en metode som finner det forhåndsbestemte antall klustre, uten en subjektivt utvelgelse av observasjoner til hvert kluster. Jeg vil analysere og implementere ulike metoder for å oppnå dette og teste dette både på datasett fra Sørlie *et al.* (2003) og på simulerte data.

Initielt har jeg sett på en metode som Tibshirani *et al.* (2001) har foreslått for å estimere antallet klustre i et gitt datasett. Ideen bak denne metoden er å teste validiteten av klustre i datasettet i forhold til klustre fra et datasett basert på en nullmodell. Metoden beregner likheten mellom individer i klustre basert på en gitt klustringsalgoritme og kommer med forslag til optimalt antall klustre basert på hvliket antall klustre som gir mest markant forskjell mellom klustring av reelle data og nullfordelingsdata.

Jeg har implementert en versjon av gapobservatoren på bakgrunn av beskrivelsen gitt i artikkelen til Tibshirani *et al.* (2001). Metoden er vurdert med hensyn til stabilitet og robusthet, ved å teste den på reelle og simulerte datasett.

Videre har jeg diskutert og testet ulike tilpasninger av gapobservatoren for å automatisk kunne oppnå samme resultat som Sørlie *et al.* (2003) fant ved sine kombinasjoner av klustring og biologisk vurdering. Jeg kom da fram til at gapobservatoren ofte velger et mindre antall klustre enn det som begrunnes av Sørlie *et al.* og av den grunn har jeg foreslått en ny versjon av gapobservatoren. Denne versjonen er en rekursiv utgave og ser ut til å resultere i et antall klustre som ligger nærmere det antall klustre som presenteres i studiet til Sørlie *et al.* (2003). Denne metoden er derfor vurdert nærmere.

# Forord

Arbeidet med denne oppgaven har pågått i to år (januar 2006 - november 2007) og jeg har i løpet av disse årene hatt kontinuerlig kontakt med to veiledere. Jeg vil derfor rette en stor takk til Knut Liestøl og Ole Christian Lingjærde for å ha bidratt i givende diskusjoner og rettledet i oppgavens fremdrift på en motiverende måte.

Takk til Gruppen for Bioinformatikk ved Institutt for Informatikk for et hyggelig miljø og for ukentlige inspirerende seminarer.

Jeg vil takke familie og venner for inspirasjon og støtte på alle områder. Dette har gjort studietiden til en god og minnerik periode. En spesiell takk til deg Karin, for å ha ventet med armene åpne, også etter lange krevende dager.

Oslo,
oktober 2007

Espen Solberg

# Innhold

# Kapittel 1

# Introduksjon

I denne oppgaven presenteres metoder for å analysere genekspresjondata. Disse genekspresjonsverdiene er hentet fra mikromatriseforsøk på brystkrefttumorer, i tillegg vil det bli simulert slike datasett. Datasettene blir brukt i studier av metoder for å finne grupperinger av individer. Et mål er da å sammenlikne mine resultater med tidligere publiserte resultater.

Denne oppgaven omhandler klustring av mikromatrisedata. Hovedproblemstillingen er estimering av antall klustre i et datasett og identifikasjon av subtyper innen et sett med observasjoner.

Klustring er en form for gruppering for å samle observasjoner etter et gitt kriterium. Et slikt kriterium er vanligvis avstand eller likhet, så en klustring av et sett med observasjoner kan derfor svare til å samle observasjoner som er mest like. Oppgaven fokuserer mest på én klustringsmetode og ser på hvordan denne metoden kan brukes sammen med en statistisk estimator for å estimere antall klustre i et datasett. Jeg vil se om dette kan idenfisere subtyper i datasettene for så å sammenlikne dette med tidligere publiserte resultater.

For lettere å kunne forstå de biologiske dataene beskriver jeg i kapittel 2 det grunnleggende om de biologiske prosessene som påvirker det menneskelige genomet. I samme kapittel forklares oppbygningen av et mikromatriseforsøk. Kapittel 3 ser på klustring og de ulike målene for å se på likhet mellom observasjoner i en klustringssituasjon. I tillegg følger en diskusjon av problemet med høye dimensjoner og en beskrivelse av prinsipalkomponent-analyse.

I kapittel 4 beskriver jeg generelle forhold rundt genekspresjonsdatasett fra mikromatriseforsøk og beskriver de datasettene som blir brukt i de følgende analysene og metodene. Disse datasettene er hentet fra et mikromatriseforsøk

publisert senest av Sørlie *et al.* (2003). Jeg sier også litt generelt om brystkreft.

Et mål med oppgaven er å se på metoder for å analysere store datamengder og i kapittel 5 beskrives i detalj metoden som er mest sentral i oppgaven; gapobservator-metoden. I dette kapittelet viser jeg resultatet fra denne metoden på datasettet fra studiet Sørlie *et al.* (2003) og diskuterer resultatet i forhold til resultatene i studiet.

I kapittel 6 testes gapobservatoren (Tibshirani *et al.,* 2001) på ulike simulerte datasett og testresultatene diskuteres. Eksempelvis simulerer jeg rundt fem sentroider og varierer spredningen av simulerte individer for å finne en grense for hvor stor spredning metoden tåler. Kapittel 7 diskuterer en ny metode basert på gapobservatoren for å gå dypere inn for å finne et representativt antall klustre i et gitt datasett. Metoden testes på Sørlie-datasettet og resultatene fra disse testene gis sammen med en diskusjon i forhold til studiet Sørlie *et al.* (2003). Kapittel 8 tester denne metoden og gapobservator-metoden på ulike simulerte datasett, og diskuterer og sammenlikner resultatene.

Kapittel 9 gir en diskusjon av utviklingen i oppgaven og gjengir noen av de viktige resultatene presentert tidligere. Her blir det også foreslått muligheter for videre arbeid med materialet fra oppgaven.

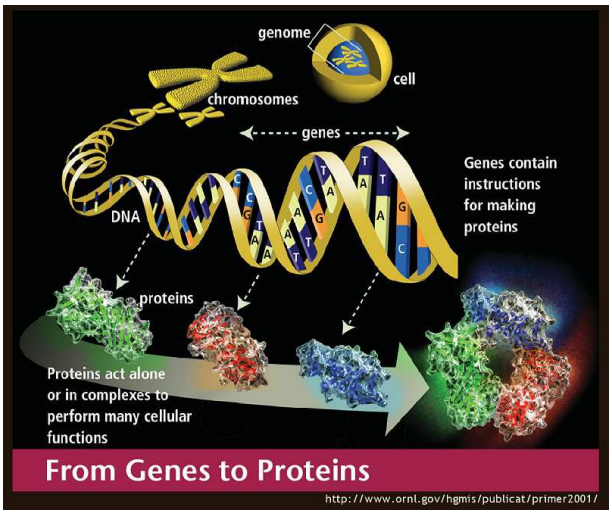
Jeg har lagt ved sentrale deler av den implementerte kildekoden og kort kommentert hver funksjon, i kapittel 10.

# Kapittel 2

# Biologisk bakgrunn

## 2.1 Det sentrale dogmet

Enhver organisme inneholder en mengde biologisk informasjon som er nødvendig for å bygge opp og vedlikeholde organismen. Denne informasjonen er spesifisert i organismens genom (Moore, 2007). Det menneskelige genomet består av DNA (deoksyribonukleinsyre) og DNA er genenes byggesteiner. Et gen kan defineres som en enhet av arvelig informasjon om det systemet en organisme utgjør, eller sagt på en annen måte, et gen er en funksjonell og fysisk enhet av intruksjoner om organismens oppbygning som arves fra foreldre til avkom (NHGRI, http://www.genome.gov/).




Figur 2.1: Fra gen til protein (Human Genome Program, http://www.ornl.gov/).

Det er slått fast at det menneskelige genomet inneholder titusener av gener. Antallet er problematisk å finne eksakt, men det er estimert til å være mellom 30 000 og 40 000 (Brown, 2007) og det spekuleres nå i at dette tallet er enda lavere.

De menneskelige genene lagres hovedsaklig som arvestoff i cellekjernene i form av DNA. Her finnes det i nesten alle tilfeller 23 kromosompar, der de første såkalte autosomene er navngitt fra 1 til 22, mens det 23. kromosomparet, kjønnskromosomene, enten er et par av såkalte X kromosomer eller et par av et Y kromosom og et X kromosom (se for eksempel Brown, 2007). I tillegg finnes det DNA i cellenes mitokondrier. Den største delen av det menneskelige genomet består av gener som ikke koder for proteiner. Kun omkring 1.5% av det menneskelige DNA består av protein-kodende gener (Human Genome Project, http://www.ornl.gov/).

Selv om vi har en definisjon av gener, er det allikevel ikke alltid enkelt å finne disse i genomet. Vi vet at et gen er en enhet med viktige funksjoner, men vi vet ikke presist hvordan vi teknisk skal skille et gen fra de deler av genomet som ikke koder for noe gen. I tillegg finnes det andre faktorer som gjør det vanskelig å lokalisere et gen. Eksempler kan være at gener kan overlappe hverandre og at det finnes ulike såkalte spleisevarianter. Det er derfor også vanskelig å bestemme eksakt hvor mange gener som finnes i en organisme.

DNA'ets primære rolle var lenge antatt å være å produsere proteiner via såkalt mRNA (se nedenfor), men det er etterhvert blitt klart at RNA også spiller en selvstendig rolle i cellen. Fordi svært mange biologiske prosesser i en organisme utføres av og med RNA og proteiner, har DNA en meget sentral rolle i en organisme. Informasjonsflyten fra DNA via mRNA til proteiner, altså fra genomet via transkriptomet til proteomet, har fått navnet *det sentrale dogmet* (Draghici, 2003).

### 2.1.1 Oppbygning av DNA

DNA er dannet fra fire ulike byggesteiner, kalt nukleotider. Disse fire nukleotidene, Adenin (A), Cytosin (C), Guanin (G) og Thymin (T), består alle av et sukkermolekyl, en fosfatgruppe og en base. Det er kun basen som skiller de ulike nukleotidene. DNA-molekyler finnes naturlig i cellene som par av kjeder med disse nukleotidene i en unik rekkefølge.

Disse kjedene er tvunnet i en såkalt dobbelheliks. Langs en slik dobbelheliks ligger nukleotidene i komplementære par. Det vil si at hver av nukleotidene kun

kan stå parallelt ovenfor én annen type nukleotid. Disse parene er A-T og C-G. Vi kan derfor ut fra den ene DNA tråden, vite hvordan den andre ser ut.

Beskrivelsen av den genetiske sammensetningen til en organisme, kalles for organismens genotype (Brown, 2007).

### 2.1.2 Transkripsjon

Nukleotidene forekommer i komplementære par. Dette gjør blant annet at cellen kan bruke den komplementære tråden som mal hvis det skjer et brudd eller en annen feil (Hirano, 2005). Forskjellige celler kopierer ulike mengder og ulike sekvenser av DNA ettersom hva slags proteiner cellen produserer. Under replikasjon blir DNA'ets dobbelheliks tvunnet opp og splittet i dette sekvensområdet ved hjelp av blant annet enzymene DNA topoisomerase og DNA helikase (Brown, 2007). For at cellen skal kunne frakte ut informasjonen i denne sekvensen, blir det produsert en komplementær sekvens i en prosess kalt transkripsjon. I denne prosessen leses og oversettes nukleotider og komplementære nukleotider bindes sammen. Dette gjøres ved hjelp av blant andre enzymet RNA-polymerase. Oppstrøms for genet finner vi sammensetninger av nukleotider som kalles promotere. Disse vil tiltrekke, binde seg til og klargjøre RNA-polymerase, andre proteiner og selve gensekvensen før avlesing av DNA.

Den nye produserte sekvensen er komplementær til DNA og kalles *messenger*-RNA (mRNA). Den er bygget opp av fire nukleotider, der tre av disse er de samme som i DNA. RNA har istedenfor T en nukleotid kalt Uracil (U). Dette vil si at hvis nukleotiden A leses av DNA tråden, vil den transkriberte mRNA tråden bestå av en U.

### 2.1.3 Translasjon og etterprosessering

Cellen har nå produsert mRNA som fungerer som en budbringer. Denne transporteres ut av cellekjernen, ut i cytoplasmaet og til ribosomene. På vei dit skal mRNA gjennom en etterprosessering som hovedsaklig består av tre underprosesser. Først skal mRNA spleises, som er å fjerne ikke-kodende deler av sekvensen, såkalte introner, og så sette sammen de kodende delene av sekvensen, kalt exoner. Hvert enkelt exon kan variere i lengde fra noen hundre til ca tusen basepar (Lander *et al.,* 2001).

I tillegg kan vært gen spleises på ulike måter. Det vil si, noen exoner kan utelates og den nye sekvensen kan da bestå av ulikt antall exoner. Vi sier da at

genet har ulike spleisevarianter, som gjør at hvert gen kan produsere flere typer proteiner.

Deretter skal det legges på en G i starten av sekvensen (capping) og det skal legges på en hale av A'er på slutten av sekvensen (polyadenylering).

Når det etterprosesserte mRNA når frem til ribosomene skal det oversettes til aminosyrer. Denne oversettingen gjøres i grupper på tre og tre nukleotider. Disse gruppene kaller vi kodoner. Denne ribosomale prosessen kalles translasjon. Resultatet er en streng av aminosyrer - et protein. Denne strengen vil raskt folde seg i en spesiell romlig struktur som er spesielt tilpasset den biologiske funksjonen proteinet har.

### 2.1.4 Celleaktivitet

Cellenes egenskaper bestemmes av hvilke proteiner de produserer, og dette bestemmes av genene og cellens omgivelser. Det finnes en rekke regulatoriske gener som styrer hvilke gener som er aktive i cellen. Disse genene fungerer som gradvise brytere som kan senke og heve mengden gener som skal kopieres og dermed også som brytere for produksjonen av ulike proteiner. Dette fungerer på den måten at de regulatoriske genene koder for proteiner som igjen regulerer andre geners aktivitet. Disse proteinene kalles transkripsjonsfaktorer. Altså styrer omgivelsene til cellen og disse regulatoriske genene egenskapene til cellene. Dette styres i normale celler veldig nøyaktig.

## 2.2 Mutasjoner og kreft

Selv om cellenes proteinproduksjon reguleres veldig nøye og arvestoffet tas godt vare på av cellene, kan det skje forandringer av gensekvensene og genuttrykket, og derfor cellens aktivitet. Det finnes en rekke ulike måter disse forandringene kan skje på. Eksempler på dette er amplifikasjoner. Når DNA'et er amplifisert vil genomet inneholde flere kopier av et DNA-segment enn originalen. Det motsatte kan også skje, altså en delesjon. Det vil si at det finnes færre kopier av et DNA-segment i genomet enn normalt.

Vi bruker oftest uttrykket kopitall når vi referer til antallet kopier av et DNA-segment. Forandringer i kopitall kan skje i både kodende og ikke-kodende deler av DNA, og det er ikke alltid en kopitallendring vil ha noen effekt på fenotypen. Det vil si, på det synlige uttrykket av genotypen.

Et annet eksempel på forandringer av gensekvenser kan være translokasjoner. Det vil si at deler av gensekvenser er byttet om og flyttet til andre steder på kromosomet. (Se figur 2.2 på neste side.)

Vi skiller hovedsaklig mellom to typer forandringer av gensekvenser. Den første er forandringer som skjer i kjønnsceller og dermed kan arves i generasjoner. Den andre er forandringer som skjer i vanlige celler (somatiske celler) og som kan forårsake kreft.

Artsmangfoldet i naturen er bygget opp gjennom mutasjoner i kjønnsceller. Dette har skjedd opp gjennom millioner av år. Vi vet fra definisjonen av gen gitt over, at et gen er arvelig informasjon om det systemet en organisme utgjør (NHGRI). I løpet av disse millionene av år har systemer utviklet seg ved mikroevolusjon, altså små endringer av genotyper som overføres til neste generasjon (Campbell & Reece, 2002). Disse små feilene blir altså arvet i form av gener i generasjoner og forårsaker det vi kaller evolusjon. Dette har gjort at jorden har blomstret opp med et mangfold av bakterier, planter og dyrearter.

Et interessant forskningsfelt innenfor denne evolusjonslæren er å finne ut av hvem og hva jordens organismer stammer fra. Dette kalles fylogeni. Innenfor dette feltet er det blant annet aktuelt å se på en type mutasjoner kalt SNP'er (Single Nucleotide Polymorphisms). Dette er individuelle forskjeller i enkeltnukleotider i genomet og blir ofte kalt punktmutasjoner. Disse finnes omtrent på hvert 1000. basepar spredt rundt i genomet og arves ofte sammen med gener. Det er derfor interessant å se på disse mutasjonene for å finne genetiske likheter hos ulike arter.

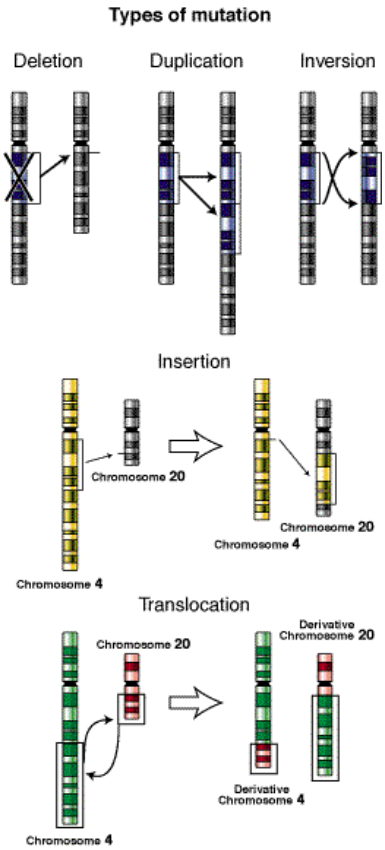
Forandringer av DNA kan skje i alle prosesser som omhandler arvestoffet. Brown skiller mellom to typer forandringer av genomet. Det første er mutasjoner, som nevnt ovenfor påvirker genomet relativt lokalt, i forhold til rekombinasjon, som påvirker store deler av kromosomer ved at en del av et kromosom bytter plass med en del av et annet kromosom. Rekombinasjoner og mutasjoner kan begge defineres som *prosesser som forandrer genomet*, men de påvirker altså genomet i ulik grad (Brown, 2007).

Mutasjoner kan skje under hele cellens levetid ved påvirkning av ulike stimuli, men de fleste mutasjoner, og alle rekombinasjoner, skjer ved kopiering av DNA eller transkripsjon av DNA.

I tillegg til at mutasjoner og rekombinasjon forandrer genomet, kan det skje feil i transkripsjonen eller sekvenskopieringen. Dette kan for eksempel skje ved at enkelte baser bytter plass, det kan skje ved feil i translasjonen av mRNA til et

protein, eller det kan skje ved feil i etterprosesseringer. Et eksempel på dette kan være feil i formingen av proteinets romlige struktur, tertiærstrukturen. På den måten vil ikke proteinet, for eksempel et enzym, passe inn i den arbeidsoppgaven den har. *Kugalskap* er et eksempel på en sykdom som antas å være knyttet til en slik feil.

Dette viser bare en liten del av de feil som kan oppstå og som faktisk skjer. Det er derfor livsviktig at cellen har ulike reparasjonssystemer som forhindrer og reparerer feil. Feil med slike prosesser er derfor veldig alvorlig. Disse feilene kan gjøre at cellen dør, eller at prosesser i cellen forandres til det unormale. Slike celler kan da utvikles til kreftceller.




Figur 2.2: Ulike mutasjoner (NHGRI), figuren gjenspeiler ikke riktig størrelsesforhold på mutasjonen i forhold til kromosomene.

### 2.2.1 Kreftceller

En årsak til at kreftceller utvikles er mutasjoner som gjør at aktiviteten til en celle endres. Det kan være mutasjoner i gener som er involvert i sentrale prosesser i cellen, som for eksempel differensiering. Med differensiering menes prosessen der en celle utvikler seg fra en stamcelle til en celle med en bestemt funksjon. Mutasjoner kan også skje i gener som er involvert i prosesser for cellevekst eller i prosesser som har med reparasjon av DNA å gjøre (Forus *et al.,* 2001). Mutasjoner i gener som har med disse prosessene å gjøre kan føre til at cellen oppfører seg på en måte som kroppen ikke gjenkjenner som riktig. Dette kan kroppens immunforsvar prøve å fjerne, men det lykkes ikke i alle tilfeller. Kreftcellene kan i mange tilfeller lure immunforsvaret fordi de opptrer mye likt en vanlig celle. Problemet er at kreftcellene ødelegger funksjoner i kroppen og noen ganger kan de også spre seg til andre deler av kroppen, kalt metastase (NLM, http://ghr.nlm.nih.gov/). Et vanlig kjennetegn på en kreftcelle er at den kopierer seg raskt, og det vil derfor også være vanskelig for immunforsvaret å lokalisere og angripe disse kreftcellene tidlig nok.

Vi kan se at det ofte er ødelagte sentrale gener som er årsak til at det dannes kreftceller. Disse sentrale genene er da ofte gener som koder for transkripsjonsfaktorer (Forus *et al.,* 2001). Endringer og avvik i disse genene kan derfor være et tegn på en kreftcelle og hvilke egenskaper denne kreftcellen har eller mangler.

Det finnes i hovedsak tre typer gener som er sentrale i kreftutvikling (Sandvik, 2001). To av disse typene gener er tumorsuppressorgener og onkogener. Disse genene koder for proteiner som er viktig i prosesser for cellevekst og celledeling. Den første for kontroll av cellevekst og celledeling, mens den andre koder for proteiner som kan øke celleveksten og celledelingen. Feil som påvirker disse genene kan da føre til økt kopitall, altså en amplifikasjon, i cellen og dette er et viktig tegn på hissige kreftceller. Den siste typen er reparasjonsgener, som koder for proteiner som reparerer arvestoffet. Skader på denne typen gener kan føre til at det blir vanskelig eller umulig å reparere mutasjoner eller skader på arvestoffet.

## 2.3 Genekspresjon

Vi kan med dagens teknologi analysere cellen og hente ut informasjonen om proteinproduksjonen i cellen. Dette kan vi gjøre på hovedsaklig to måter. Det første er å analysere mRNA som gir oss informasjon om transkriptomet og det andre er å analysere de produserte proteinene som vil gi oss informasjon om

proteomet.

Den mest brukte av disse metodene er den første (Jenssen, 2002). Den ser på genekspresjonen i en celle. Produktet av genekspresjon er transkriptomet, og det er det vi måler. Transkriptomet defineres som mengden av RNA fra proteinkodende gener, altså mRNA, som befinner seg i en celle på et gitt tidspunkt (Brown, 2007). Når vi ser på genekspresjonsverdiene i en celle må vi være klar over at disse er ulike fra celletype til celletype, fordi ulike celler har ulike arbeidsoppgaver. De er også ulike fra en gitt tid til en annen, fordi cellen kan være utsatt for ulikt stress fra tid til annen og reagerer ulikt på dette.

Når vi henter ut mRNA ser vi kun på de transkriberte sekvensene og ikke på sluttproduktet og vi kan ikke være sikre på hvilke proteiner et gitt mRNA koder for. Dette er, som nevnt tidligere, fordi både mRNA og proteiner kan etterprosesseres til flere varianter og dermed til ulike proteiner.

Den andre metoden analyserer sluttproduktet direkte, nemlig proteinene. Ved å se på proteinene i cellen kan vi hente ut mer presise data, fordi vi da vet akkurat hvilke proteiner cellen produserer og mengden av disse. Denne metoden er derimot vanskelig å bruke for å analysere hele proteomet.

### 2.3.1 Mikromatriser

Analysen og ekstraheringen av genekspresjon kalles transkriptomanalyse. Det finnes flere ulike metoder for å hente ut data om en celles arvestoff og genekspresjon, men dagens forskning har størst fokus på bruken av såkalte mikromatriser i denne typen analyse.

Det finnes flere ulike typer mikromatriser. To av de mest brukte metodene er *in situ* syntetisering (for eksempel Affymetrix) og cDNA spotting (Lockhart & Winzeler, 2000; Jenssen, 2002; Draghici, 2003). *In situ* synteser bruker en rekke oligonukleotider som forsøksmateriale i hvert eksperiment. Det vil si entrådet DNA som er mellom 5 og 50 nukleotider langt (NCBI, http://www.ncbi.nlm.nih.gov/). Disse syntetiseres på matrisen nukleotid for nukleotid. Denne varianten av mikromatriser blir ofte kalt oligonukleotidmatriser og produserer mer presis data enn cDNA spotting. Metoden er nå også utviklet for å teste mange gener hos mange individer.

cDNA spotting baserte matriser ble utviklet tidligere enn *in situ* matriser. Allerede i 1995 ble den første artikkelen som rapporterte bruk av cDNA matriser publisert (Stanford Medicine Magazine, http://mednews.stanford.edu/). På den tiden var oligomatrisene kun på forskningsstadiet, mens de idag benyttes i stør-

re grad enn cDNA matriser.

Jeg vil i fortsettelsen skrive om cDNA mikromatriser fordi artiklene jeg har tatt utgangspunkt i har brukt denne metoden. Jeg vil henvise til denne metoden når jeg omtaler mikromatriser.

Mikromatriser er et laboratorieverktøy som kan gi oss forståelse av arvestoffet basert på tilgang til et stort antall DNA sekvenser.

En mikromatrise er en glassplate som skal få festet til seg entrådete gener i et matrisemønster, altså et tabellmønster. Før et mikromatriseforsøk utføres må det planlegges nøye hvilke gener som skal analyseres, hvor mange ganger et gen skal være tilstede på matrisen, hvilken rekkefølge de skal festes på matrisen, hvilke individer som skal være med, osv. Selve designet av matrisen er en viktig del av forsøket. Dette er viktig fordi det finnes mange feilkilder som vil gi støy på målingene og vi vil alltid prøve å få så presise data fra forsøket som mulig.

Dataene vi får ut av dette verktøyet er altså genekspresjon i et forhold mellom to vevstyper, en referanse og en prøve. Det er vanlig å se på forholdet mellom ulike kreftvev og en blandet prøve fra cellelinjer hentet fra ulike kreftceller. En cellelinje er flere celler som i et laboratorie er fremstilt fra én celle. Verdiene får vi da i en matrise $M$:

$$M = \log_2 \left( \frac{G_{\text{prøve}}}{G_{\text{referanse}}} \right) ,$$

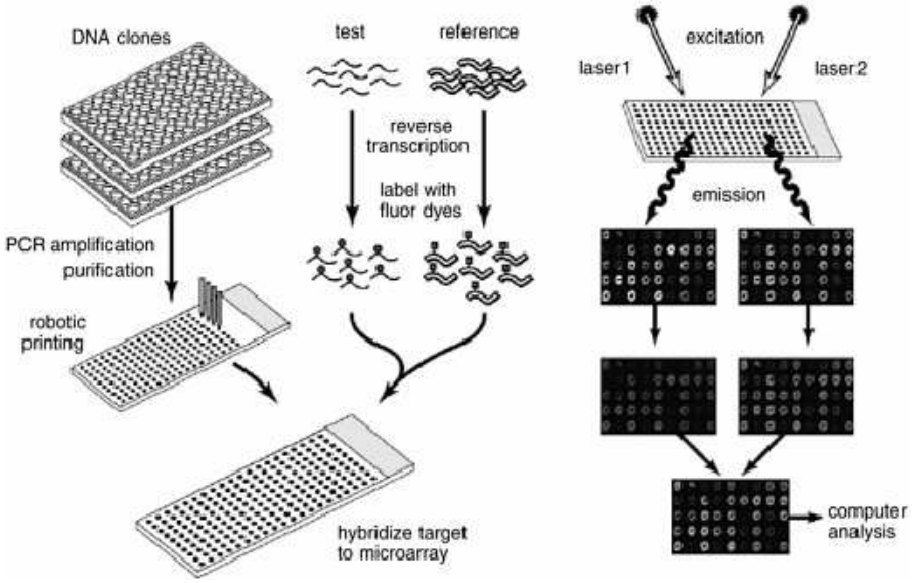
der $G_{\text{prøve}}$ er genekspresjonsverdiene fra prøven og $G_{\text{referanse}}$ er genekspresjonsverdiene fra referansen.

Det finnes også en annen mye brukt mikromatrisemetode som ikke analyserer genekspresjon, men ser på kopitallet i cellen. Et unormalt kopitall sier noe om instabiliteten til det genetiske utrykket. Dette kan derfor brukes til å gi en indikasjon på kreft og på hva som forårsaket kreft.

### 2.3.2 Design og feilkilder

Vi vil at et mikromatriseeksperiment skal være nøyaktig og presist. Med det menes at målingene skal representere det vi ønsket å måle. Ofte vil det i et eksperiment være data som fraviker eller skiller seg fra resterende data. Slike målinger kalles uteliggere. Uteliggere er data det må tas hensyn til fordi de kan ha mye å si for den statistiske modelltilpasningen. I noen tilfeller kan uteliggere ødelegge den statistiske modelleringen, mens i andre tilfeller kan uteliggere være det vi ser etter i et eksperiment.

Det finnes en mengde feilkilder som kan gjøre at et eksperiment får et dårlig




Figur 2.3: Figuren viser i korte trekk hvordan et cDNA mikromatriseeksperiment blir utført (The National Academic Press)

resultat (Kerr & Churchill, 2001; Draghici, 2003; NIH, http://discover.nci.nih.gov/). Feilkildene gir støy (inklusivt uteliggere), og støy kan oppstå i alle faser av et eksperiment. Dette kan være eksperimentelle urenheter, unøyaktigheter, numeriske feil, med mer. Normalisering av data, som har som mål å kompensere for systematiske skjevheter i resultatene, kan også bidra til feilmengden. Under normalisering prøver vi å få data fra mikromatrise-eksperimentet over på en standard form, hvor det er mulig å gjøre sammenlikninger mellom gener og prøver/individer.

Når vi tester titusener av gener vil det være vanskelig å modellere fordi vi har mange målinger og hver av målingene vil inneholde litt støy. Mengden støy vil bli mer dominerende ettersom antallet målinger er stort. Det er altså vanskelig å gjøre nøyaktige og presise målinger av så mange gener. Vi har normalt et større antall variable, i dette tilfellet gener, enn vi har individer eller observasjoner. Det er derfor vanskelig å finne gode statistiske modeller for slike eksperimenter.

Andre kilder til feil vil oppstå fordi vi har med biologiske systemer å gjøre. Det er vanskelig å oversette en datamatrise med genekspresjonsverdier direkte til biologien i cellen og vice versa, både fordi et genuttrykk kan gi flere ulike proteiner og fordi det ligger mye annen informasjon bak en tallverdi i matrisen. Dette kan eksempelvis være informasjon om tilstanden til individet. Vanskelig-

heter som dette kan bidra til feilmengden i selve datasettet i tillegg til å prege våre slutt-tolkninger av eksperimentet og gjøre disse mer unøyaktig. Våre slutt-tolkninger blir da unøyaktig fordi vi oversetter mikromatrisedata og numeriske tabeller til biologisk fakta, noe som i seg selv er ikke-trivielt.

En annen feilkilde skyldes bruken av fargestoffer i eksperimentene. Disse fargestoffene blir ikke tatt opp i like mengder, som gjør at resultatet kan bli skjevt. Det finnes en rekke andre feilkilder i tillegg til de nevnte og disse kan forskere redusere i stor grad ved å nøye og gjennomtenkt bestemme eksperimentets design.

Eksempler på ulike mikromatrise designtyper er referansedesign, 'dye-swap' design og 'loop' design. Referansedesign er et mikromatriseeksperiment som bruker den samme referansen til flere ulike prøver. Dette brukes som oftest når eksperimentet går over lengre tid og utføres på flere laboratorier. *Dye-swap* design brukes for å korrigere skjevheter i fargeopptak. Dette gjøres ved at både referanse og prøve bruker begge farger. *Loop* design kan gjøres på ulike måter, men prinsippet er at de ulike prøvene inntar roller som både referanse og prøve. Dette foregår da i et sirkulært mønster, eller ved at alle prøver syntetiseres med alle andre prøver. Dette skal øke sannsynligheten for å finne forskjeller (NuGO, http://www.nugo.org/).

### 2.3.3 Replikering, randomisering og blokking

Uansett valg av design, finnes det i følge Draghici tre hovedprinsipper som bør følges. Disse tre er replikering, randomisering og blokking (Draghici, 2003). Med replikering menes at en oppgave eller et eksperiment repeteres flere ganger. Dette gjør at vi i et mikromatriseeksperiment kan kartlegge feil. Ved å analysere et gen flere ganger, det vil si å feste samme gen flere steder på matrisen, vil vi kunne danne et grunnlag for å sile ut uteliggere og å gjøre målingene mindre avhengig av lokasjonen på matrisen. Hvis vi da bestemmer oss for å replikere et gen, må vi bestemme hvor mange replikater vi vil ha med og hvor disse skal festes på matrisen i forhold til hverandre. Draghici sier at det er best å spre slike replikeringer på samme matrise. Grunnen til det dette, er at alle feltene på en matrise er med i samme forsøk og vil dele mange feilkilder. Forklaringen til hvorfor man bør spre eksperimenter, er for at hvert felt med samme gen ikke skal oppleve samme lokale forstyrrelser.

Hovedpoenget med replikering er å redusere variansen (Draghici, 2003). Det er vist at eksperimentresulater blir merkbart bedre ved bruk av 3 replikater iste-

denfor ingen replikater av hvert gen på en mikromatrise (Lee *et al.*, , 2000).

Neste prinsipp Draghici nevner er randomisering. Det vil si at vi skal tilfeldiggjøre faktorer i eksperimentet. Dette skal gjøre at den statistiske modelleringen blir enklere og det vil minske effekten av konfunderende faktorer. Det vil si minske effekten av avhengigheter mellom faktorer i ekseperimentet (Moore & McCabe, 2003). Et eksempel på randomisering er å tilfeldige plassere replikater av gener på en matrise.

Det siste prinsippet Draghici tar opp er blokking. Prinsippet går ut på at de støyfaktorene som er tilstede i et eksperiment vil være konstante innenfor en blokk (batch). Innenfor en blokk vil målinger være mer homogene enn målinger fra ulike blokker (Tempelman, 2005). Et typisk eksempel på en blokk er mikromatrisen selv. Innenfor mikromatrisen vil forholdene være veldig like fordi hele mikromatrisen i et eksperiment blir behandlet samtidig.

Anta at hver blokk har et skifte i nivå som er avhengig av hvilken blokk de befinner seg i, altså en effekt av typen

$$x_{ijk} = \ldots + a_j + \ldots \quad ,$$

for et datasett $X = (x_{ijk})$ der $j$ er blokknummer og $i$ og $k$ er henholdsvis individnummer innenfor hver blokk og gennummer. Vi kan da enkelt korrigere for den blokk-avhengige faktoren $a_j$ ved å foreta en blokk-sentrering. Vi estimerer $a_j$ ved

$$\hat{a}_j = \frac{1}{n_j p} \sum_{i,k} x_{ijk}$$

hvor $n_j$ er antall individer i blokk $j$ og $p$ er antall gener. Deretter erstatter vi $x_{ijk}$ med $\tilde{x}_{ijk} = x_{ijk} - \hat{a}_j$.

# Kapittel 3

# Klustring

Etter analysen av et mikromatriseeksperiment sitter man igjen med en numerisk tabell med logaritmiske forhold mellom genekspresjoner. En slik tabell er arrangert som en matrise. Når man ser på to vektorer $\mathbf{x}$ og $\mathbf{y}$ fra denne matrisen, kan disse enten være gener (prober) eller individer (tumorprøver). Hvis hver vektor representerer et gen vil vi da ha en vektordimensjon lik antall individer. Hvis hver vektor derimot representerer et individ, vil vi ha en vektordimensjon lik antall gener.

I mikromatriseforsøk vil antall gener normalt være langt større enn antall individer. Typiske tall kan for eksempel være ca 100 individer og 40 000 prober. Med prober menes ofte gener, men med det forbehold at et gen kan representeres i flere prober. I fortsettelsen vil jeg for enkelhets skyld bruke betegnelsen gener både når jeg snakker om ulike gener og om prober.

En slik matrise består altså av en veldig stor datamengde, og vektorene i matrisen kan være av veldig høy dimensjon. Det er derfor vanskelig å trekke ut informasjon direkte fra matrisen ved å bruke vanlige statistiske metoder. Man gjør da ofte en klustring av dataene for å finne likheter og ulikheter. En klustring er en gruppering av observasjoner og observasjonene kan både representere individer og gener. Man kan også bruke andre metoder, som for eksempel prinsipalkomponent-analyse (PCA, se avsnitt 3.4 på side 26.)

Jeg bruker i denne oppgaven klustring for å gruppere individer. Jeg vil da gruppere sammen individer som har stor similaritet, altså kategorisere. Det har blitt gjort mange studier på slik kategorisering av genekspresjonsdata fra krefttumorer og dette har blant annet resultert i en subkategorisering av kjente krefttyper (Sørlie *et al.,* 2001; Sørlie *et al.,* 2003). Man skiller da ut grupper av individer med liknende genekspresjoner. Klustring av gener derimot gjøres ofte for å iden-

tifisere typiske genekspresjonsmønstre og for å samle gener med liknende funksjon eller koregulering (Lingjærde, 2005). Det er også vanlig å klustre individer og gener samtidig, og disse klustringene er da uavhengig av hverandre. En visualisering av en slik klustring kan gi et oversiktlig bilde på likheter og ulikheter mellom individer og mellom gener. (Se figur 3.1 på neste side.)

Det finnes flere ulike klustringsmetoder. Overordnet skiller vi mellom hierarkisk klustring og ikke-hierarkisk klustring. Eksempler på dette er *agglomerativ* og *divisiv* hierarkisk klustring og de ikke-hierarkiske metodene *k-means* og *self-organizing maps* (SOM).

## 3.1 Avstandsmål og similaritetsmål

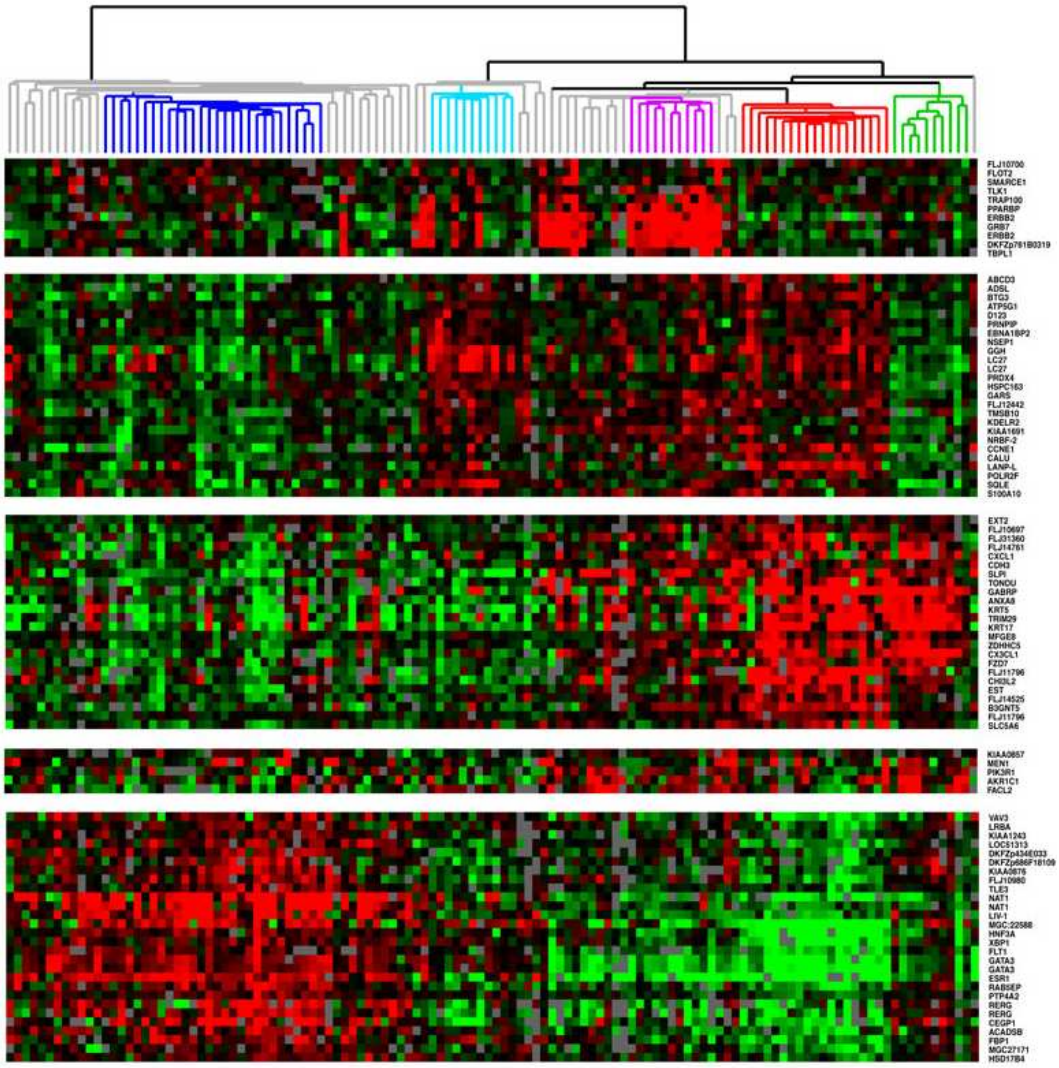
For å måle avstandene mellom de ulike grupperte punktene brukes et mål for sammenlikning. Eksempler på slike mål er *Euklidsk avstand*, *Minkowski avstand*, *Manhattan avstand*, *Chebychev avstand* og *Pearson korrelasjon*. Her er det sistnevnte et similaritetsmål, mens de foregående er avstandsmål. De numeriske likhetsverdiene vil bli brukt som input til klustringsalgoritmene. Selve klustringsresultatet kan i betydelig grad påvirkes av hvilket avstandsmål eller similaritetsmål som brukes.

Ett similaritetsmål er korrelasjon, som ser på likhet. Avstandsmål derimot er funksjoner som regner ut avstanden mellom to punkter. En slik funksjon tar to punkter i $\mathbb{R}^n$ og produserer en skalar. Et avstandsmål skal ha tre egenskaper (Draghici, 2003; Huang *et al.,* 2003):

- **Symmetri**: For $x, y \in \mathbb{R}^n$ er $d(x, y) = d(y, x)$.

- **Positivitet**: For $x, y \in \mathbb{R}^n$ er $d(x, y) \geq 0$ og $d(x, y) = 0 \Leftrightarrow x = y$.

- **Trekantulikhet**: For $x, y, z \in \mathbb{R}^n$ er $d(x, y) \leq d(x, z) + d(z, y)$.
  Dette innebærer at det ikke er mulig å korte ned veien fra $x$ til $y$ ved å gå innom er tredje punkt $z$.



Figur 3.1: Hierarkisk klustring av genekspresjonsdata fra brystkrefttumorer. Klustringen er gjort på 122 individer (her i kolonner) og 540 gener, men det er bare vist de genene som varierer mest blant individene (Sørlie *et al.,* 2003). Individer som er gruppert til ulike subkategorier av brystkreft er gitt de fem fargene mørk blå, lys blå, lilla , rød og grønn, mens individer farget i grått ikke er gruppert til noen subkategori.

### 3.1.1 Eksempler på mål for avstand

Anta i fortsettelsen at vi har gitt to vektorer $\mathbf{x} = (x_1, ..., x_n)^T \in \mathbb{R}^n$ og $\mathbf{y} = (y_1, ..., y_n)^T \in \mathbb{R}^n$. *Euklidsk avstand* mellom $\mathbf{x}$ og $\mathbf{y}$ er gitt ved

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad . \tag{3.1}$$

Dette avstandsmålet brukes ofte og til mange praktiske formål.

*Manhattan avstanden* mellom $\mathbf{x}$ og $\mathbf{y}$ er

$$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \quad . \tag{3.2}$$

I dette tilfellet finnes det flere veier av samme lengde mellom to punkter, korteste vei er altså ikke unik.

*Minkowski avstand* er en generalisering av *Euklidsk avstand* og *Manhattan avstand*. *Minkowski avstanden* mellom $\mathbf{x}$ og $\mathbf{y}$ er

$$d(\mathbf{x},\mathbf{y}) = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \quad , \tag{3.3}$$

der $p \geq 1$ = orden.

Det vil si at for *Euklidsk avstand* er $p = 2$. Ved å øke $p$, vil vi da øke følsomheten for parvise komponenter med stor differanse.

Et annet avstandsmål er *Chebychev avstand*, som er definert som $d(\mathbf{x},\mathbf{y}) = \max_i |x_i - y_i|$.

Hvis det er ønskelig å vektlegge ulike retninger forskjellig, kan *Mahalanobis avstand* være et godt avstandsmål. Dette avstandsmålet er definert som $d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T S^{-1} (\mathbf{x}-\mathbf{y})}$, der $S$ er en $n \times n$ symmetrisk og positiv definit matrise og $(\mathbf{x}-\mathbf{y})^T$ er den transponerte til $\mathbf{x}-\mathbf{y}$.

### 3.1.2 Eksempel på mål for similaritet

Cosinus til vinkelen mellom to vektorer er et mål på similaritet mellom vektorene. Hvis vektorene skaleres likt på alle akser vil ikke dette påvirke vinkelen. Gitt $n$-dimensjonale vektorer $\mathbf{x} = (x_1, ..., x_n)^T \in \mathbb{R}^n$ og $\mathbf{y} = (y_1, ..., y_n)^T \in \mathbb{R}^n$ tilfredsstiller vinkelen $\theta$ mellom vektorene relasjonen

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|} \quad , \tag{3.4}$$

$$\text{der } \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

$$\text{og } \|\mathbf{x}\| \quad = \quad \sqrt{\sum_{i=1}^{n} x_i^2} \quad .$$

Hvis vektorene $\mathbf{x}$ og $\mathbf{y}$ sentreres om 0 før vinkelen beregnes, får vi *Pearson korrelasjonen* mellom $\mathbf{x}$ og $\mathbf{y}$.

$$s(\mathbf{x},\mathbf{y}) \quad = \quad \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}}$$

Vi kan se at $s(\mathbf{x},\mathbf{y})$ ligger i intervallet $[-1,1]$ for alle $\mathbf{x}$, $\mathbf{y}$, ved å ta utgangspunkt i (3.4) og Cauchys ulikhet $|x \cdot y| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\|$:

$$|\cos(\theta)| \overset{(3.4)}{=} \frac{|x \cdot y|}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} \overset{\text{Cauchy}}{\leq} \frac{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} \quad = \quad 1$$

og derfor har vi at $-1 \leq \cos(\theta) \leq 1$.

## 3.2 Klustringsmetoder

Resultatet av å klustre et sett med vektorer er at hver vektor blir tilordnet en gruppe. Dette kan representeres numerisk og noen ganger også visuelt. For eksempel, for hierarkisk klustring vil den visuelle presentasjonen av resultatet være et dendrogram (en trestruktur). Klustringen er allikevel bare ett element i en mikromatrise analyse (Draghici, 2003). Et annet element er tolkningen av resultatet satt inn i en biologisk sammenheng.

Det er mulig å få mange ulike resultater og presentasjoner fra de forskjellige klustringsmetodene. Jeg vil her gi en oversikt over noen ikke-hierarkiske og hierarkiske klustringsmetoder.

### K-means klustring

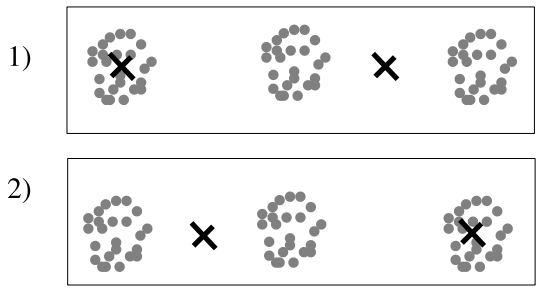
*K-means* er en av de enkleste og raskeste algoritmene og derfor også mye brukt (Draghici, 2003). *K-means* er en ikke-hierarkisk eller en såkalt *flat* metode. Metoden fungerer på den måten at det initielt settes $k$ tilfeldige klustersentre. Antallet

klustre er ofte bestemt på bakgrunn av erfaringer. Disse kan for eksempel plasseres helt tilfeldig i verdi-rommet eller tilfeldig i områdene med flest punkter (Draghici, 2003). Man gjør da gjerne klustringen flere ganger for til slutt å velge den klustringen som har minst variasjon om de endelige klustersentrene. En annen metode er å initielt sette klustersentrene lenger fra massemiddelpunktet (tyngdepunktet fra alle dimensjoner) enn de gitte punktene og langt fra hverandre. Etter initialiseringen startes en løkke:

1. Regn ut avstandene fra hvert klustersenter til hvert punkt
2. Tilordne hvert punkt til nærmeste klustersenter
3. Regn ut nytt klustersenter som massemiddelpunktet mellom klustrenes punkter
4. Hvis ikke et termineringskriterium er tilfredsstilt, gå til punkt 1

Det er vanlig at løkka termineres når den har nådd et maksimalt antall iterasjoner eller når klustrene har stabilisert seg, det vil si når ingen punkter endrer klustertilhørighet.

Når det gjelder initialplasseringen av klustersentrene er det verdt å merke seg noen spesielle tilfeller. To kjøringer av metoden, med ulik initialplassering, kan gi ulikt resultat. Hvis datasettet som brukes har tydelige klustre, vil resultatet derimot bli det samme hver gang, forutsatt at antall klustersentre er likt antall tydelige klustre. I tilfeller der vi har tydelige klustre og antallet tydelige klustre ikke er likt antall klustersentre, vil vi kunne få ulikt resultat ved gjentatte kjøringer. Et eksempel på dette kan sees i figur 3.2.

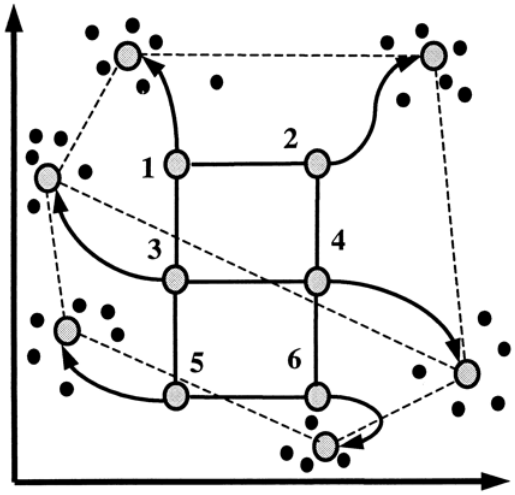



Figur 3.2: Figuren viser ulike resultater fra to mulige *k-means* klustringer der punktene danner tre tydelige klustre og antallet klustersentre er 2. De grå sirklene viser punkter (individer), mens de svarte kryssene er klustersentre.

I ekstreme tilfeller kan metoden også produsere klustre uten noen tilhørende punkter. Det er derfor viktig å nøye bestemme antall klustre og å gi klustersentrene en fornuftig initiell plassering. En mulig heuristikk kan være å plassere klustersentrene tilfeldig i områder som ser ut til å tilhøre et kluster. Dette vil gjøre at det ikke produseres tomme klustersentre, men det vil også sette en begrensing på metodens objektivitet. Resultatet vil da være påvirket og kan påvirkes i den retning det ønskes (Draghici, 2003).

### Self-Organizing maps

Klustringsmetoden *Self-Organizing Maps* (SOM) (Kohonen, 2001) starter med å definere et rutenett, vanligvis i en, to eller tre dimensjoner. Det vil si som henholdsvis en linje, et parallellogram (se figur 3.3) eller en parallellpiped. Dette rutenettet består av noder som blir til klustersentre. (På figuren er dette nodene 1-6.) Antallet noder og typen rutenett må bestemmes på forhånd.




Figur 3.3: Self-Organizing Maps: Her vises verdi-rommet med nodene 1-6 (grå sirkler) som er klustersentre. Pilene viser hvordan disse nodene migrerer til å bli et senter i hvert kluster. De små svarte prikkene er punkter (Tamayo *et al.,* 1999).

Nodene har en *referansevektor* som projiseres ned på verdi-rommet. (I eksempelfiguren er verdi-rommet $\mathbb{R}^2$, mens vektor-rommet, for eksempel ved klustring av individer, er lik antall gener.) Metoden går så gjennom en løkke:

1. Velg et tilfeldig punkt
2. Beregn avstand mellom referansevektorene og det tilfeldig valgte punktet
3. Flytt noden som er nærmest dette punktet i verdi-rommet nærmere punktet og flytt de nærliggende nodene etter, men disse i mindre grad.
4. Hvis et termineringskriterium ikke er nådd, gå til punkt 1.

Metoden har ofte et termineringskriterium som gjør at metoden går ut av løkka når punktene flyttes mindre enn en forhåndsbestemt verdi. Punktene blir så tilordnet de nodene (klustersentrene) som ligger nærmest og klustringen er ferdig.

Det at noder som ligger nærme den valgte noden i punkt 3 blir flyttet i samme retning, gjør at nærliggende noder er mer like hverandre enn noder lengre fra hverandre. Derfor har nodene (altså klustrene) et naboforhold med de nodene (klustrene) som de har en kant til.

Det er avgjørende for resultatet av metoden hvor mange klustersentre som velges, hvilken dimensjon og hva slags rutenett som velges. Disse faktorene bestemmes derfor på grunnlag av erfaring (Draghici, 2003).

### Hierarkisk klustring

Hierarkisk klustring har som mål å produsere en trestruktur kalt et dendrogram (se figur 3.4 på side 24). Dette gjøres ved å ta utgangspunkt i en tabell med avstander mellom alle klustre eller grad av similaritet mellom alle klustre. Tabellen blir regnet ut ved hjelp av avstandsmålene og similaritetsmålene som ble presentert i avsnitt 3.1 på side 16. Jeg vil i det følgende referere til denne tabellen som en avstandstabell, selv om den ikke nødvendigvis viser avstand.

Når vi måler avstander mellom klustre, er det hovedsaklig en av følgende tre metoder (*inter-klustermål*) som blir brukt. La $R = \{r_1, \ldots, r_n\}$ og $S = \{s_1, \ldots, s_m\}$ være to klustre og la $d(R, S)$ angi avstanden/similariteten mellom $R$ og $S$:

- **Single linkage**: Måler minste avstand eller største similaritet mellom punkter i klustre. Det vil si at det måles mellom de punktene fra hvert kluster som er nærmest hverandre.

$$\text{For avstandsmål:} \qquad d(R,S) = \min_{x \in R, y \in S} (dist(x,y))$$

$$\text{For similaritetsmål}: \quad d(R,S) = \max_{x \in R, y \in S}(likhet(x,y))$$

- **Complete linkage**: Måler største avstand eller minste similaritet mellom punkter i klustre. Det vil si at det måles mellom de punktene fra hvert kluster som er lengst fra hverandre.

$$\text{For avstandsmål}: \quad d(R,S) = \max_{x \in R, y \in S}(dist(x,y))$$
$$\text{For similaritetsmål}: \quad d(R,S) = \min_{x \in R, y \in S}(likhet(x,y))$$

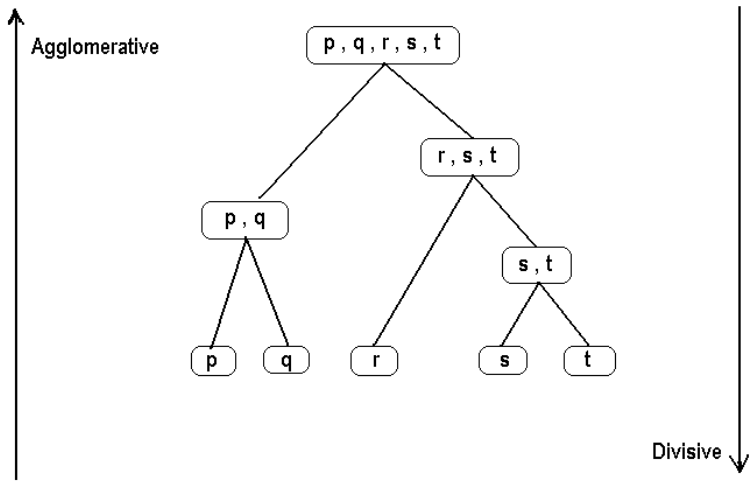
- **Average linkage**: Måler gjennomsnittlig avstand eller similaritet innen hvert kluster og sammenstiller de ulike klustrene med denne oppnådde verdien. Velger da klustre med minst gjennomsnittlig avstand eller størst gjennomsnittlig similaritet.

$$\text{For avstandsmål}: \quad d(R,S) = \frac{1}{n_R n_S} \sum_{x \in R} \sum_{x \in S} dist(x,y)$$
$$\text{For similaritetsmål}: \quad d(R,S) = \frac{1}{n_R n_S} \sum_{x \in R} \sum_{x \in S} likhet(x,y)$$

Det finnes hovedsaklig to hierarkiske klustringsmetoder (se figur 3.4 på neste side). Den første metoden arbeider fra topp til bunn og kalles divisiv. Metoden starter med alle punkter i ett kluster og deler så opp dette klusteret i flere klustre i henhold til avstandstabellen. Terminering av metoden skjer når alle punkter er i hvert sitt kluster. Det har da blitt produsert et dendrogram.

Den andre hierarkiske klustringsmetoden arbeider fra bunn til topp og kalles agglomerativ. Denne metoden fungerer på følgende måte. Anta at vi har $n$ punkter:

1. Legg disse $n$ punktene i $n$ klustre.
2. Regn ut en avstandstabell med alle avstander mellom par av klustre.
3. Slå sammen de to klustrene som er nærmest/mest like.
4. Fortsett punkt 2 og 3 til kun ett kluster gjenstår.




Figur 3.4: Figuren viser to forskjellige hierarkiske klustringsmetoder. Agglomerative metoder starter med hvert punkt i et eget kluster og slår sammen klustre til alle punkter er i samme kluster. Divisive metoder starter med alle punkter i samme kluster og deler opp dette klusteret til alle punkter er sitt eget kluster (Resampling Stats, http://www.resample.com/).

Metoden har nå produsert et dendrogram. Resultatet av dette er avhengig av hvilket inter-klustermål som er brukt.

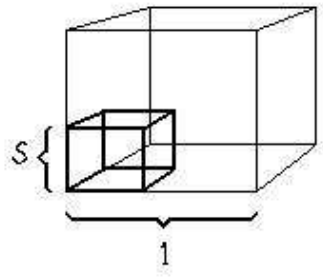
Dersom vi ut fra et dendrogram ønsker å dele i et bestemt antall klustre, kan vi enkelt velge en terskel (et cut-off nivå) i passende høyde. Det å velge denne terskelen kan være komplisert. Den kan velges på bakgrunn av ønsket antall klustre, ønsket høyde eller ut fra en mengde heuristikker. Allikevel er det som oftest ikke riktig å bare velge en terskel. Forholdene er mer kompliserte, så det er derfor viktig å finne metoder for å skille komponentene i mer informative klustre.

## 3.3 Dimensjonsproblemet

Et datasett fra mikromatriseeksperimenter kan for eksempel være i størrelsesorden $500 \times 100$, der antall gener (rader) er mye større enn antall individer (kolonner). (Se avsnitt 4.1 på side 28.) I et slikt tilfelle er det veldig vanskelig å forestille seg hvordan data ligger i rommet. Det viser seg at jo flere dimensjoner (her gener) datasettet består av, jo lenger vil observasjonene (individene) være fra tyngdepunket av alle observasjonene (Hastie *et al.,* 2001).

Anta en enhetskube i $p$ dimensjoner og at vi velger en underkube med side-

lengde $s$. Da er volumet av underkuben $s^p$. Hvis data finnes vil kun en liten del av dataene ligge i underkuben når $s < 1$ og $p$ er stor. En klassifikasjon i et slikt tilfelle vil derfor være veldig usikker.




Figur 3.5: En enhetskube og en underkube med sidelengder $s < 1$.

En annen konsekvens er, som nevnt, at nesten alle punktene i et utvalg ligger nær ytterkanten av det området det velges fra (Hastie *et al.,* 2001). Andelen punkter nær ytterkantene øker og nærmer seg 100% når dimensjonen øker.

Anta at $n$ datapunkter er uniformt trukket i en $p$-dimensjonal enhetskule med senter i 0. Den euklidske medianavstanden fra origo til nærmeste datapunkt er da (Hastie *et al.,* 2001):

$$d(p,n) = \left(1 - \left(\frac{1}{2}\right)^{\frac{1}{n}}\right)^{\frac{1}{p}}$$

Eksempler:

$$\begin{aligned} d(10,100) &= 0.6080 \\ d(50,100) &= 0.9053 \\ d(500,100) &= 0.9901 \end{aligned}$$

Dette resulterer i at den euklidske avstanden mellom punktene blir større i høyere dimensjon. Vi kaller dette for dimensjonsproblemet. I et mikromatrisetilfelle vil dette kunne gjøre en god klassifisering av punkter vanskeligere. Dette kan blant annet motivere bruken av prinsipalkomponent-analyse, som beskrives i avsnitt 3.4 på neste side.

## 3.4 Prinsipalkomponent-analyse

Prinsipalkomponent-analyse (PCA) brukes i behandlingen av store og flerdimensjonale datasett, når dataene er korrelerte (Speed, 2003). PCA brukes for å redusere dimensjonen (antall kolonner) i datasettet uten at informasjonskvaliteten blir betydelig redusert. I mikromatrisesammenheng har vi et datasett som består av mange flere variable (gener) enn observasjoner (individer). Ved å la kolonnene i datasettet representere gener, kan vi benytte PCA til å redusere dimensjonen av gener. Dette kan være ønskelig for å redusere redundansen i datasettet og for å muliggjøre tilpasning av statistiske modeller som ellers er vanskelig å bruke på høydimensjonelle data. Ved å la kolonnene i datasettet representere individer kan vi benytte PCA til å redusere dimensjonen av individer, noe som kan være nyttig for å finne grupper av relaterte individer.

PCA går ut på å finne en eller flere spesielle lineærkombinasjoner av kolonnene i datasettet. Disse lineærkombinasjonene blir valgt slik at variansen maksimeres. De dimensjonene som blir beholdt er da de som bidrar mest til variansen i datasettet.

Det brukes ofte matriseprogrammer for å utføre beregningene (Lay, 2005). Metoden fungerer på følgende måte (Lay, 2005). Anta at vi har en $n \times p$ matrise $A$ med ekspresjonsverdier for $n$ individer og $p$ gener. Vi finner prinsipalkomponentene på følgende måte:

- Beregn gjennomsnittsekspresjonsverdien for hvert gen:

$$\bar{x} = \frac{1}{n}\mathbb{1}^T A$$

- Sentrer variablene (kolonnene) i A:

$$A_c = A - \mathbb{1}\bar{x} = (I_n - \frac{1}{n}\mathbb{1}\mathbb{1}^T)A \ .$$

- Regn ut kovariansmatrisen:

$$S = \frac{1}{n-1}A_c^T A_c \ .$$

  Merk at S er en $p \times p$ matrise.

- Finn egenverdiene $\lambda_1 \geq ... \geq \lambda_p$ til kovariansmatrisen $S$ og de tilhørende egenvektorene $v_1, ..., v_p$.

- Prinsipalkomponentene er nå gitt ved:

$$p_j = A_c v_j \qquad , j = 1, \ldots, p$$

Ved å velge de dimensjonene med mest variasjon, det vil si prinsipalkomponentene i sekvensen $p_1, p_2, \ldots$, kan den opprinnelige genekspresjonsmatrisen representeres i færre dimensjoner. Velger vi for eksempel de $k$ første prinsipalkomponentene, erstattes $A$ av

$$\tilde{A} = [p_1, \ldots, p_k] = A_c V_k \ \in \mathbb{R}^{n,k} \quad ,$$

hvor $V_k = [v_1, \ldots, v_k]$. Antall kolonner er da redusert fra $p$ til $k$ og det viser seg at disse komponentene er resultatet av en ortogonal regresjon på dataene (Lay, 2005).

Metoden PCA regner ut verdier i hver dimensjon som minimerer kvadratsummen av de ortogonale avstandene. På denne måten er resultatet den beste tilnærmingen til den opprinnelige genekspresjonsmatrisen, med dimensjonene lagt ortogonalt i de retningene med størst variasjon.

PCA kan brukes sammen med klustring. Ved å først utføre PCA for så å klustre med for eksempel en av de ovennevnte klustringsmetodene, vil vi kunne gi klustringsmetodene mulighet for å danne tydeligere definerte klustre. Klustrene blir da tydeligere fordi vi har redusert antall dimensjoner med PCA og fordi PCA har laget en ny tilpasning til punktene som inneholder hovedstrukturene i den originale genekspresjonsmatrisen.

Det finnes også en egen klustringsmetode som er basert på PCA, kalt *Gene Shaving* (Hastie *et al.,* 2000).

# Kapittel 4

# Datasett

Dataene som benyttes i denne oppgaven representerer genekspresjoner i tumorer fra brystkreftpasienter. En beskrivelse av disse dataene gis nedenfor. I tillegg benyttes kunstige, simulerte data. En nærmere beskrivelse av disse er gitt i kapittelene der de opprettes og brukes.

## 4.1 Sørlie-datasettet

Datasettet er publisert i Sørlie *et al.* (2003). Det representerer genekspresjon i tumorer fra 122 brystvevprøver, målt ved hjelp av cDNA mikromatriser. Av disse prøvene er 115 maligne brystkrefttumorer (carcinomaer), 3 er godartede (fibroadenoma) og 4 er fra normalt brystvev. Det ble målt 42000 genekspresjonsverdier for hvert individ/ tumor, fra i alt 45943 ulike prober. Det vil si at alle mikromatrisene ikke inneholdt de samme genene. Grunnen til dette er at dataene hovedsaklig er hentet fra mikromatriseforsøk basert på to ulike matrisedesign (*Supporting Materials And Methods*, Sørlie *et al.,* 2003).

I alt 115 av individene som er representert i datasettet er, som sagt, carcinomaer. Med dette menes ondartede svulster som har oppstått i vevet rundt organer og som kan ha spredd seg til andre deler av kroppen (NHGRI; MedicineNet.com). I dette tilfellet er det snakk om kreft i brystvev.

De fleste av individene har lokalt avansert brystkreft, på nivå T3/T4 og N0-N2. (Se tabell *Supporting Table 2* og *Supporting Materials And Methods* fra Sørlie *et al.,* 2003.) T3/T4 betyr at tumorene har vokst seg relativt store. N0 betyr at kreften ikke har spredt seg til nærliggende lymfenoder, mens N1/N2 betegner at kreften har spredt seg til et begrenset antall nærliggende lymfenoder (Breastcancer.org,

http://www.breastcancer.org/).

Sørlie *et al.* valgte å studere 552 såkalte *intrinsic* prober. Dette settet representerer 534 ulike gener, det vil si at noen gener ble representert med flere prober. Genene som ble valgt hadde ekspresjonsverdier som varierte mye mellom tumorer, men varierte lite mellom målinger gjort fra samme tumor før og etter 15 uker med kjemoterapi. (Informasjon om individene og tumorene finnes i tabellen *Supporting Table 2* fra Sørlie *et al.* (2003).)

### 4.1.1 Valg av 5 klustre

Sørlie *et al.* (2003) tiordnet ut fra klusteranalyse individene i fem grupper (se figur 3.1 på side 17). For hver gruppe regnet de ut en sentroide med utgangspunkt i de individene som hadde høyest korrelasjon med hverandre. Hver sentroide representerte en tentativ subtype av brystkreft. Et nytt individ kan da klassifiseres til en av subtypene ved å finne den sentroiden som korellerer mest med ekspresjonsvektoren til individet. De fem subtypene ble kalt:

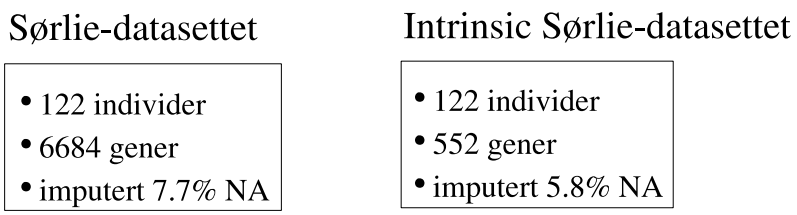
- Luminal A ($p_1 > 0.32$)
- Luminal B ($p_2 > 0.28$)
- Errb2+ ($p_3 > 0.34$)
- Basal ($p_4 > 0.41$)
- Normal-like ($p_5 > 0.31$)

Tallene i parentes angir det opprinnelige inkluderingskriteriet som ble benyttet for å avgjøre hvilke individer som skulle inngå ved beregning av sentroidene. Dette inkluderingskriteriet var en grense for korrelasjon mellom sentroidene og individene for inkludering i de ulike klustrene. Ekspresjonsvektoren til et individ måtte ha en korrelasjon med den respektive sentroiden som oversteg tallet gitt i parentes. Det betyr at ikke alle individene i datasettet var med for å bestemme sentroidene (se figur 3.1).

### 4.1.2 Data jeg har brukt

Sørlie-datasettet er lastet ned fra Stanford Microarray Database (SMD, http://genome-www5.stanford.edu/). *Skjermdumper* fra denne nedlastingen er tilgjengelig (Skjermdumper, http://heim.ifi.uio.no/) I hovedtrekk velges kun gener med 80% data

som ikke er flagget. Flagget data skyldes upålitelige målinger eller målinger med en viss mengde støy. Datasettet inneholder 7.7% manglende verdier (NA) som har vært flagget. Det består da av 6684 gener og 122 tumorprøver (individer).




Figur 4.1: Oversikt over datasettene jeg har benyttet. Manglende verdier betegnes i figuren som NA = not available.

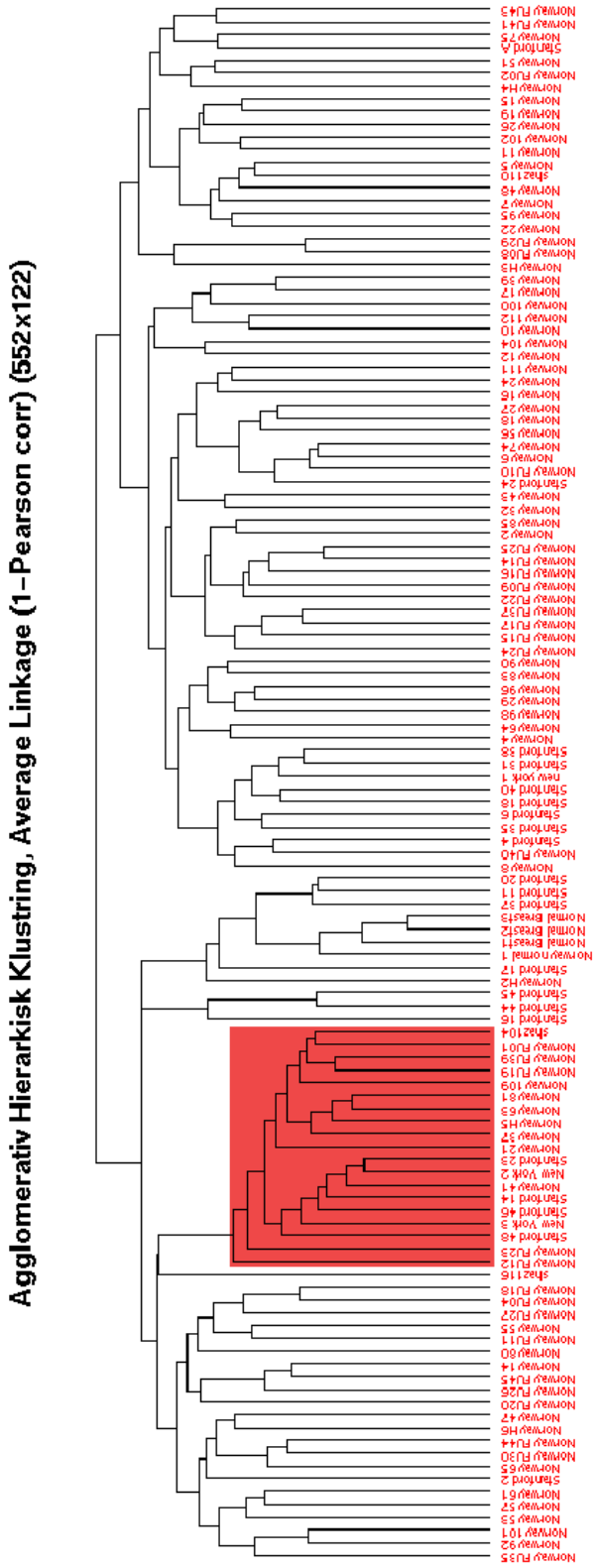
Videre er data imputert ved hjelp av pakken *impute.knn* i statistikk-programmet *R* (R, http://www.r-project.org/), som beregner en imputert verdi for de manglende verdiene på bakgrunn av de 10 nærmeste naboer. Dette datasettet kalles i det følgende for *Sørlie-datasettet*.

Jeg har også laget et datasett med *intrinsic* gener. Dette datasettet består av 122 individer og 552 prober. Disse probene er de samme som Sørlie *et al.* (2003) kaller *intrinsic*. Dataene har 5.8% manglende verdier, som er imputert i *R* på samme måte som beskrevet over. Figur 4.2 på side 32 viser et dendrogram av dette *intrinsic* datasettet. Det er dette datasettet som er mest brukt i de følgende testene jeg har gjort og vi refererer til dette datasettet som *intrinsic Sørlie-datasettet*.

Dendrogrammet i figur 4.2 er et forsøk på å reprodusere dendrogrammet i figur 3.1 på side 17. Det finnes flere grunner til at de to dendrogrammene ikke er helt like. For det første er det en egenskap ved slike dendrogrammer at klustrene kan speiles uten at selve klustringen blir annerledes. Dette gjør at to helt like klustringer kan representeres ulikt. I tillegg er det gjort forskjellige imputeringer og brukt forskjellige klustringsmetoder i de to figurene. Figur 3.1 bruker Eisens klustringsalgoritme (HybridHclust, http://www.r-project.org/) som bruker et inter-klustermål kalt centroid linkage og inneholder en egen imputeringsmetode, mens figur 4.2 bruker average linkage og imputeringsmetoden nevnt over. En annen grunn til ulikhet mellom figurene, er at data brukt i figur 4.2 er hentet fra SMD som krever flere valg for nedlasting av data. Det kan derfor være små forskjeller i hvilke dataverdier som er brukt og hvilke normaliseringer som er

gjort på dataene.

Allikevel er klustringsresultatet veldig likt. Ved å se på individene som er klustret sammen i de to figurene, har jeg funnet at det er stor overenstemmelse. Individene i klustrene som i figur 3.1 er farget med en annen farge enn grå, er i stor grad de samme individene som er klustret sammen i figur 4.2. Eksempelvis er alle individene fra figur 3.1 sitt røde kluster (*basal-like*) også tydelig klustret sammen i figur 4.2. Jeg har markert dette klusteret med rød bakgrunnsfarge i denne figuren.




Figur 4.2: Dendrogram fra klustring av intrinsic Sørlie-datasettet, produsert i *R*. Individene markert over rød bakgrunnsfarge tilhører klusteret *Basal-like*. De samme individene finnes igjen i figur 3.1 på side 17 der det samme klusteret også er markert rødt (*Supplement Table 2 og 3*, Sørlie *et al.*, 2003).

# Kapittel 5

# Bestemme antall klustre med gapobservatoren

Noen klustringsmetoder (som k-means) forutsetter at man på forhånd angir antall klustre, mens andre (som hierarkisk klustring) ikke gjør det. I begge tilfeller kan det være ønskelig å estimere det reelle antallet klustre i et datasett.

Tibshirani *et al.* (2001) har foreslått en metode for å estimere antall klustre i et datasett. Metoden er basert på en observator de har kalt *gapobservatoren* og fungerer ved å ta utgangspunkt i en klustringsmetode. Ved å se på similariteten eller avstanden mellom punkter innenfor klustre, og iterativt øke antall klustre, vil metoden foreslå et optimalt antall klustre.

## 5.1 Beskrivelse av gapobservatoren

Løst sagt er et optimalt antall klustre $k^*$ det antall grupper av observasjoner som distinkt skiller seg ut på bakgrunn av et valgt similaritetsmål. For å estimere $k^*$ brukes ofte metoder basert på en funksjon $W_k$ som for en gitt oppdeling av punkter i klustre angir innen-gruppe variansen, eller mer generelt innen-gruppe spredningen. Denne funksjonen baserer seg på et gitt avstandsmål.

Anta at vi har data $X = (x_{ij})$ for individ $i = 1, \ldots, n$ og gen $j = 1, \ldots, p$ der vi ønsker å klustre individer. Dette vil også fungere i tilfeller der vi ønsker å klustre gener og vi kan da bruke den transponerte til datatrisen $X$. Betrakt videre $d_{ii'}$

som avstanden mellom individ $i$ og $i'$. For kvadrert euklidsk avstand får vi da

$$d_{ii'} = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2 \quad .$$

Anta at individene er gruppert i $k$ klustre $C_1, \ldots, C_k$, der $C_r$ er individene i kluster $r$ og $n_r = |C_r|$. Summen av de parvise avstandene mellom individene i kluster $r$ blir da

$$D_r = \sum_{i,i' \in C_r} d_{ii'} \quad .$$

Den totale normerte innen-gruppe spredningen kan da uttrykkes som

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r \tag{5.1}$$

Anta at vi ønsker å se på grupperinger av individene i opptil $K$ klustre. Da regner vi ut $W_k$ for $k = 1, \ldots, K$. Vi får da et sett med innen-gruppe spredninger $\{W_1, \ldots, W_K\}$ som normalt avtar i verdi for økende $k$.

Anta at det finnes et optimalt antall klustre $k^*$. Vi ønsker da å finne et estimat $\hat{k}$ på det optimale antall klustre. Anta i det følgende at klustrene vi finner når $k = k^*$, svarer til de optimale klustrene.

- For $k < k^*$ vil det da være observasjoner som er klustret sammen som ideelt sett ikke skulle vært det, fordi de tilhører ulike optimale klustre, og vi forventer derfor at $W_{k+1} < W_k$. Verdien av $W_k$ vil da avta merkbart for økning av $k$.

- For $k > k^*$ vil det finnes minst ett kluster som er delt unødvendig. Det vil si at minst ett optimalt kluster er delt i to klustre. Vi forventer stadig at $W_{k+1} < W_k$ , men at verdiforandringen ikke vil være av like betydelig størrelse som for $k < k^*$.

- For $k = k^*$ er ingen av de optimale klustrene oppdelt i flere klustre og det er da heller ikke tilfelle at to optimale klustre ligger i samme kluster. I denne situasjonen har vi $W_k << W_{k-1}$ og får da et estimat $\hat{k}$ for $k^*$. (Operatoren $<<$ brukes for å understreke vesentlig mindre enn.) Vi kan da også se at et plott av $W_k$ vil gi et *knekk* i kurven for $k = k^*$.

Vi finner altså $\hat{k}$ som den laveste $k$, der vi observerer at $W_k$ avtar mest.

Gapobservatoren, *Gap(k)*, sammenligner $\log W_k$ med en forventet $\log W_k$ som er regnet ut fra simulert data uniformt fordelt over et passende definert utfallsrom til dataene. Tibshirani *et al.* (2001) nevner to metoder for generering av referansedata. Den ene av disse metodene tar hensyn til formen på fordelingen av originaldataene ved å trekke referanseobservasjoner fra fordelingene innenfor prinsipalretningene.

Gitt originaldata $X = (x_{ij})$ bestående av $n$ rader og $p$ kolonner der radene skal klustres, danner vi $B$ referansesett $Z_b$ , $b = 1, \ldots, B$:

**Metode I**

*For hver* $j = 1, \ldots, p$ *trekkes verdier* $z_{1j}, \ldots, z_{nj}$ *fra den uniforme fordelingen* $U[a_j, b_j]$, *der* $a_j = \min_i(x_{ij})$ *og* $b_j = \max_i(x_{ij})$. *Sett* $Z_b = (z_{ij})$.

**Metode II**

*Først sentreres kolonnene* $j = 1, \ldots, p$:

$$x_{ij}^{new} = x_{ij} - \bar{x}_j \ , \ i = 1, \ldots, n \quad ,$$

*der* $\bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$. *Finn så singulærverdi-dekomposisjonen til* $X^{new}$:

$$X^{new} = U \, D \, V^T \ .$$

*La*

$$X' = X^{new} \, V = U \, D \ .$$

*For hver* $j = 1, \ldots, p$ *trekkes verdier* $z_{1j}, \ldots, z_{nj}$ *fra den uniforme fordelingen* $U[a_j, b_j]$, *der* $a_j = \min_i(x'_{ij})$ *og* $b_j = \max_i(x'_{ij})$. *Sett* $Z = (z_{ij})$. *Transformer så tilbake og få referansedatasettet* $Z'$ *via*

$$Z' = Z \, V^T \ .$$

*Legg så til det tidligere subtraherte gjennomsnittet for hver kolonne*

$$z_{ij}^{new} = x'_{ij} + \bar{x}_j \quad , \ i = 1, \ldots, n \quad og \quad j = 1, \ldots, p \ .$$

*Sett så* $Z_b = (z_{ij}^{new})$.

Gapobservatoren har nå et sammenlikningsgrunnlag i de $B$ referansesettene og kan brukes som en estimator for å estimere antall klustre i et datasett. Selve utregningen er beskrevet i følgende algoritme:

**Algoritme (Gapkriteriet)**

1. Velg klustringsmetode. Klustre originaldata i $k = 1, \ldots, K$ klustre og regn ut innen-gruppe spredningene $W_k$.

2. Generer $B$ referansedatasett ved å bruke en av de to metodene over. Klustre så hvert genererte datasett og regn ut $W_{kb}^*$ for $b = 1, \ldots, B$ og $k = 1, \ldots, K$. Regn ut gapobservatoren

$$Gap(k) = \frac{1}{B} \sum_{b=1}^{B} \log\left(W_{kb}^*\right) - log\left(W_k\right) .$$

3. La

$$\bar{l}_k = \frac{1}{B} \sum_{b=1}^{B} \log\left(W_{kb}^*\right) ,$$

og regn ut standardavviket

$$sd_k = \sqrt{\frac{1}{B} \sum_{b=1}^{B} \left(\log\left(W_{kb}^*\right) - \bar{l}_k\right)^2} .$$

Definer $s_k = sd_k\sqrt{1 + 1/B}$. Velg så det optimale antall klustre for $k = 1, \ldots, K$ via kriteriet

$$\hat{k} = argmin_k \left\{Gap(k) \geq Gap(k+1) - s_{k+1}\right\} . \tag{5.2}$$

## 5.2 Implementasjon av gapobservatoren

Jeg har implementert gapobservator-algoritmen som en funksjon i Matlab (http://www.mathworks.com/). Denne funksjonen har jeg kalt GapObs. (Se avsnitt 10.1 på side 71 for kildekode.)

Jeg har brukt metode II på side 35 for å beregne referansedatasett og agglomerativ hierarkisk klustring med average linkage for å klustre individene. Avstandsmålet jeg har brukt er Euklidsk avstand. Dette har jeg brukt fordi resultatet er

enklere å jobbe med og enklere å forstå. Dette gjelder spesielt når jeg ser på avstanden mellom klustre i et dendrogram av klustringen. Siden jeg gjør en sentrering og normalisering før jeg kjører GapObs er dette nesten det samme som å bruke korrelasjon (se avsnitt 5.2.1).

Det er derfor viktig at data er sentrert og normalisert for hvert individ. Hvis dette ikke gjøres, vil GapObs kunne gi ufornuftige svar. Jeg gjør sentrering og normalisering via funksjonen *sentrer_rad*, som jeg har skrevet i Matlab (se avsnitt 10.3.1 på side 75.)

Jeg kan også bruke korrelasjon direkte ved å forandre en parameter til en funksjon i kildekoden. Det har ingen betydning å sentrere og normalisere før jeg regner ut korrelasjon.

Resultatet av GapObs er det estimerte antall klustre, og i tillegg listes alle klustre som oppfyller kriteriet (5.2). Det gis også et plott av gap-kurven. De parametre jeg må gi til GapObs er:

- *data*: et datasett der radene vil bli klustret
- $K$: maksimalt antall klustre som vil bli testet
- $B$: antall simuleringer av referansedatasett

Jeg har funnet én annen tilgjengelig implementasjon av gapobservatoren (Jornsten, http://www.stat.rutgers.edu/~rebecka/RCode/). Denne er implementert i *R*. Etter å ha rekodet implementasjonen fra *R* til *Matlab* slik at klustringsmetoden som er brukt er lik den jeg bruker, gir begge implementasjonene samme resultat. Min implementasjon er derimot betydelig raskere.

### 5.2.1 Likhet mellom korrelasjon og euklidsk avstand

Nedenfor viser jeg at korrelasjon er proporsjonalt med euklidsk avstand i tilfellet der data er sentrert og normalisert. Dette er aktuelt fordi Sørlie *et al.* (2003) bruker korrelasjon og jeg ønsker å bruke euklidsk avstand.

Anta at vi har en matrise $X = (x_{ij})$ med $i = 1, \ldots, n$ rader og $j = 1, \ldots, p$ kolonner. Anta videre at $X$ er radsentrert:

$$\sum_{j=1}^{p} x_{ij} = 0 \quad \forall i$$

og radene er normalisert:

$$\sum_{j=1}^{p} x_{ij}^2 = 1 \quad \forall i.$$

Korrelasjon mellom individ (rad) $i$ og $i'$ er gitt ved:

$$\begin{aligned} \text{corr}(\mathbf{x}_i, \mathbf{x}_{i'}) &= \frac{\sum_{j=1}^{p} \left(x_{ij} - x_{i\bullet}\right)\left(x_{i'j} - x_{i'\bullet}\right)}{\sqrt{\sum_{j=1}^{p} \left(x_{ij} - x_{i\bullet}\right)^2 \sum_{j=1}^{p} \left(x_{i'j} - x_{i'\bullet}\right)^2}} \\ &= \sum_{j=1}^{p} x_{ij} x_{i'j} \quad , \end{aligned} \tag{5.3}$$

der $x_{i\bullet} = \frac{1}{p} \sum_{j=1}^{p} x_{ij}$ er gjennomsnittsverdien for individ $i$.

Kvadrert euklidsk avstand mellom individ $i$ og $i'$ er:

$$\begin{aligned} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 &= \sum_{j=1}^{p} x_{ij}^2 + \sum_{j=1}^{p} x_{i'j}^2 - 2 \sum_{j=1}^{p} x_{ij} x_{i'j} \\ &= 2 - 2 \sum_{j=1}^{p} x_{ij} x_{i'j} \quad . \end{aligned} \tag{5.4}$$

Under de gitte forhold kan vi fra uttrykk (5.3) og (5.4) se at det å maksimere korrelasjon er det samme som å minimere den (kvadrerte) euklidske avstanden. Jeg kan derfor bruke euklidsk avstand og allikevel få resultater som samsvarer med de vi får for korrelasjon. Det vil gjøre det enklere å forstå resultatene av tester, samtidig som jeg kan sammenlikne mine resultater med de som er publisert Sørlie *et al.* (2003).

## 5.3 Alternative tilpasninger

Anta et tilfelle der det finnes enkeltstående punkter som ligger utenfor tydelige klustre. Slike uteliggere vil bidra mye til å øke innen-gruppe spredningene, så fremt de ikke defineres som egne klustre.

Et slikt tilfelle vil i stor grad påvirke resultatet av K-means klustring, siden antallet klustre må være gitt på forhånd. Hierarkisk klustring vil kunne håndtere et slikt tilfelle bedre ved at klustringen ikke påvirkes i like stor grad av et lite antall uteliggere. Dette skyldes at hierarkisk klustring ikke bestemmer antall klustre før klustringen, men allikevel vil uteliggere kunne forstyrre den hierarkiske klustringen og spesielt vil de forstyrre gapobservatoren i estimeringen av antall klustre.

En mulig tilpasning av gapobservatoren kan derfor være en versjon av GapObs som fjerner uteliggere. Dette kan for eksempel gjøres ved å fjerne individer som er mer enn tre standardavvik fra gjennomsnittsindividet. En annen variant er å fjerne individer som etter en kjøring av GapObs anses som uteliggere etter et gitt kriterium. Et slikt kriterium kan eksempelvis være at individer som ligger mer enn tre standardavvik utenfor sitt klustersenter markeres som uteliggere.

Andre forslag til nye tilpasninger eller forandringer av gapobservatorkriteriet:

1. Erstatt gapobservator-kriteriet i (5.2) med

$$\hat{k} = argmin_k\{\text{Gap}(k) + s_k \geq \text{Gap}(k+1) - s_{k+1}\} \tag{5.5}$$

   Motivasjonen for dette kriteriet er den erfaring at gapobservatorenofte ser ut til å velge et for lavt antall klustre. Siden gapobservatoren med kriteriet i (5.2) ofte velger et lavt antall klustre, vil kriteriet i (5.5) være strengere og derfor oftere velge et høyere antall klustre. Dette betyr at en gapobservator med dette nye kriteriet velger et antall klustre $k$ hvor $\text{Gap}(k)$ har en større økning i forhold til $k-1$, enn det som hadde vært tilfelle for gapobservatoren med kriteriet i (5.2).

   Et problem med dette kriteriet kan være at gapobservatoren i spesielle tilfeller ikke finner noen $k$ som tilfredsstiller kriteriet. Dette skjer i tilfeller der gapkurven ikke har *bratt* nok stigning for noen $k$.

2. La gapobservatoren velge det antall klustre som maksimerer gapkurven innenfor den samme toppen av gapkurven (lokalt maksimum) som ble valgt av det vanlige gapestimatet gitt ved kriteriet (5.2), det vil si:

$$\hat{k} = argmin_k\{k \geq \hat{k}^* \text{ og } Gap(k+1) \leq Gap(k)\} \quad , \tag{5.6}$$

   der $\hat{k}^*$ er det vanlige gapestimatet i henhold til kriteriet i 5.2.

Vi velger da det punktet på gapkurven som er det lokale maksimum i nærheten av det punktet som ble gitt av det vanlige gapestimatet.

3. La gapobservatoren velge det antall klustre som maksimerer gapkurven innenfor den samme toppen av gapkurven som ble valgt av kriteriet (5.2), men med tilleggskravet om at dette må gi en økning av gapkurven som er større enn en gitt faktor $\alpha > 0$:

$$\hat{k} = argmin_k\{k \geq \hat{k}^* \text{ og } Gap(k+1) - \alpha \leq Gap(k)\} \tag{5.7}$$

For $\alpha = 0$ er dette forslaget det samme som det forrige. Kriteriet velger en gapverdi som ligger i området der den lokale toppen av kurven begynner å flate ut.

4. La gapobservatoren bli valgt som det global maksimum på gapkurven:

$$\hat{k} = argmax_k\ Gap(k)\ . \tag{5.8}$$

Vi velger dette kriteriet når vi ønsker det antallet klustre som gir den minste innen-gruppe spredningen i datasettet, i forhold til innen-gruppe spredningen i referansedatasettene.

5. Kjør GapObs en gang og kjør så GapObs adskilt på hvert av klustrene fra resultatene av den første kjøringen. Jeg beskriver og tester ut denne rekursive varianten i kapittel 7.

## 5.4 Gapobservatoren brukt på Sørlie-datasettet

Jeg har brukt gapobservatoren på Sørlie-datasettet og på intrinsic Sørlie-datasettet (se kapittel 4 på side 28). Hovedfokus er å sammenlikne individer. Jeg er derfor interessert i å klustre individer, så jeg velger å bruke inputdata med individer som rader og gener som kolonner.

I det følgende har jeg valgt å teste opp til K=10 klustre og gjøre B=100 simuleringer av referansedata. Tidsbruk for en slik test på Sørlie-datasettet er omtrent 8 minutter og på intrinsic Sørlie-datasettet omtrent 11 sekunder. (Linux-maskin med 3.0GHz prosesser og 1GB internminne.)

### 5.4.1 Resultater

50 kjøringer av GapObs på Sørlie-datasettet og på intrinsic Sørlie-datasettet med $K = 10$ og $B = 100$ gir meg resultatene i tabell 5.1. Fra denne tabellen ser vi at $\hat{k} = 2$ for kjøringer på intrinsic Sørlie-datasettet, mens kjøringer på Sørlie-datasettet gir $\hat{k} = 2$ og $\hat{k} = 3$. Hvis antall simuleringer $B$ vokser, vil vi få et resultat som er entydig. Tabellen gir oss likevel en idé om at antall klustre i Sørlie-datasettet ikke er veldig tydelig bestemt.

| *datasett* $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Sørlie-datasettet | 0 | 22 | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| intrinsic Sørlie-datasettet | 0 | 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabell 5.1: Resultat-tabell fra kjøringer av GapObs. Her er $k$ antall foreslåtte klustre.

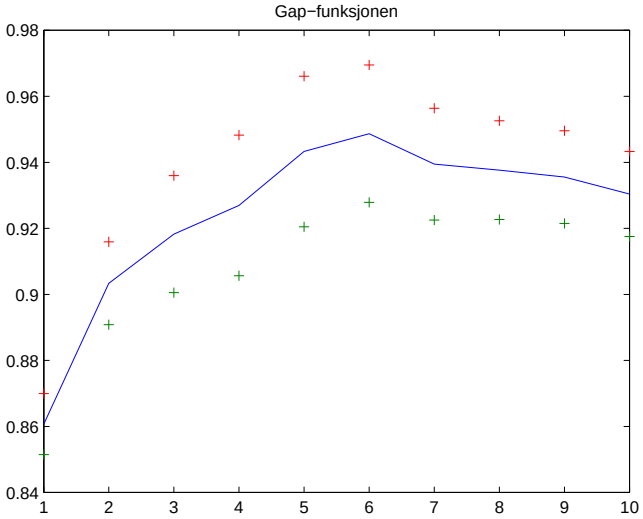
Figurene 5.1 på neste side og 5.2 på neste side viser resultatkurvene fra vilkårlige kjøringer av GapObs på hvert av de to datasettene. Plusstegnene over og under kurven er henholdsvis verdiene pluss og minus et standardavvik.

Figur 5.1 på neste side, som er fra en kjøring av GapObs på Sørlie datasettet med 6684 gener, viser at det velges 2 klustre. Dette ser vi fordi $Gap(2) \geq Gap(3) - s_3$, der $s_3$ er standardavviket for $k = 3$. Gapobservatoren sammenlikner altså punktet på kurven med plusstegnet som ligger under kurven for det neste antall klustre. Dette vil også si at $k = 10$ aldri vil bli valgt i dette tilfellet.

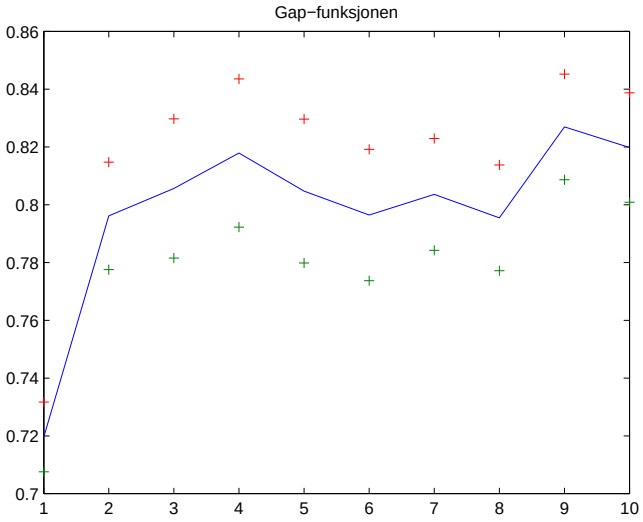
Selv om det velges to klustre, ser vi at kurven stiger helt til $k = 6$. Det vil si at hvis gapobservatorkriteriet (5.2) på side 36 hadde vært strengere, ville et høyere antall klustre blitt valgt. (For forslag til hvordan kriteriet kan gjøres strengere, se avsnitt 5.3 på side 38 om tilpasninger av gapobservatoren.)

Det gitte kriteriet (5.2) velger altså to klustre. Bak dette ligger det at hvis vi deler opp datasettet i mer enn to klustre, så vil ikke dette gi tilstrekkelig mindre innen-gruppe spredning $W_k$. Kurven måtte da enten vært brattere, eller vi måtte ha lavere standardavvik. Men, vi kan også se av figuren at marginene er små i forhold til å velge tre klustre.

Figur 5.2 på neste side, som er fra kjøring av GapObs på intrinsic Sørlie-datasettet med 552 gener, viser at det også her velges $\hat{k} = 2$ klustre. Igjen ser vi at $\hat{k}$ ikke velges som maksimum på kurven. Vi har et lokalt maksimum for $k = 4$ og et globalt maksimum for $k = 9$. I denne figuren ser vi også at kurven ikke stiger bratt nok, eller at standardavviket er for lite, til at $\hat{k}$ velges større.




Figur 5.1: Plottet viser gapkurven fra GapObs på Sørlie-datasettet. Plusstegnene viser ett standardavvik over og under hvert punkt på kurven. Resultatet fra denne kjøringen var $\hat{k} = 2$.




Figur 5.2: Plottet viser gapkurven fra GapObs på intrinsic Sørlie-datasettet. Plusstegnene viser ett standardavvik over og under hvert punkt på kurven. Resultatet fra denne kjøringen var $\hat{k} = 2$.

### 5.4.2 Diskusjon

Den biologiske artikkelen jeg har tatt utgangspunkt i (Sørlie *et al.,* 2003), indikerer ut fra en kombinasjon av klustring og biologisk vurdering at brystkreftdata grupperer seg i fem klustre. Det har vist seg at man kan gi en biologisk tolkning av disse fem klustrene. Jeg ville derfor se om det var mulig å identifisere denne grupperingen med et automatisk kriterium. GapObs er det første forsøket jeg har gjort.

Fra de tester jeg har gjort med gapobservatoren får jeg ikke de resultatene artikkelen har kommet fram til. De fem antatte klustrene synes ikke spesielt godt i gap-funksjonen. En første slutning fra dette er at data ikke bør deles i fem klustre, men at to klustre er mer fornuftig. Dette vil da kunne forklares med at det ikke finnes noen klarere grupperinger enn disse to klustrene og at individene i hvert av klustrene ikke kan skilles noe mer.

En annen slutning angående det at mine resultater ikke er i samsvar med resultatene fra artikkelen, er at gapobservatoren ikke klarer å finne sammenhengen mellom individer fordi data er i så mange dimensjoner at det er nærmest umulig å kunne se noen tydeligere grupperinger ved automatiske metoder.

Den første slutningen stiller svakt i forhold til resultatene fra Sørlie *et al.* (2003) og jeg har tro på at det finnes en tydeligere og sterkere sammenheng mellom flere individer i datasettet. Med det mener jeg at det er gapboservatoren som bør justeres til å være bedre egnet for slike datasett. Dette kan enten være en justering av kriteriet, eller det kan være en justering av fokus på de ulike dimensjonene.

En justering av kriteriet kan gjøres, men dette vil fort resultere i at valget av antall optimale klustre bli subjektiv og valgt på bakgrunn av empiri om det aktuelle datasettet. En slik justering må derfor utarbeides nøye og på et generelt grunnlag som er felles for datasett av denne typen.

En justering av fokuset på de ulike dimensjonene (genene) kan derimot være en bedre forandring. Dette kan gjøres på flere måter. En variant kan være å se på de dimensjonene der data skiller seg mest. Dette gjøres til en viss grad når vi ser på intrinsic Sørlie-datasettet, da vi kun ser på de genene som varierer mye mellom individer. Det er i hovedsak disse genene vi biologisk er mest interessert i å analysere.

For å se på de dimensjonene der data skiller seg mest, kan vi for eksempel se på de første prinsipalretningene og sjekke om individene skiller seg tydeligere. Gapobservatoren gjør noe liknende, den ser hele tiden på avstandene mellom

punktene i datasettet i forhold til punkter tilfeldig i en $p$-dimensjonal boks, som ved bruk av Metode II på side 35 er lagt i prinsipalretningene. Vi har derfor en variabel fokus på de ulike dimensjonene.

Et kjerneproblem med dette er at vi ser denne $p$-dimensjonale boksen fra veldig lang avstand. Vi har sett i resultatene fra kjøringene av GapObs at vi overordnet kan dele datasettet i to. Et oppfølgingsspørsmål blir da å spørre om vi får en finere inndeling av individene hvis vi zoom'er inn på de klustrene vi har. Vi har altså to klustre, men kan vi, ved å se på en $p$-dimensjonal boks som ligger rundt verdiområdet til et av disse klustrene, se at individene faktisk har en gruppering som først på dette nivået er blitt synlig? Dette er utgangspunktet for det jeg i kapittel 7 på side 58 beskriver som en rekursiv versjon av GapObs.

# Kapittel 6

# Testing av gapobservatoren

Jeg har gjort tester av gapobservatoren på datasett som er basert på de fem sentroidene fra Sørlie *et al.* (2003). I det studiet har forfatterne gjort hierarkisk klustring på alle individene, for så å observere at data hovedsaklig ligger i fem klustre. Fra disse fem klustrene er de punktene som er nærmest kjernen i hvert kluster tatt med i beregningen av sentroidene. Det vil si at kun de individene som har en høy korrelasjon med en gjennomsnittvektor for klusteret er tatt med. Fra disse individene er det beregnet en sentroide for hvert kluster.

Fra de 122 individene i intrinsic Sørlie-datasettet var det da igjen 79 individer som hadde tilhørighet til en av de fem sentroidene, mens 43 individer ikke var inneholdt i noe kluster.

## 6.1 Simuleringer

Data ble simulert med utgangspunkt i fem sentroider $\mathbf{c}_1, \ldots, \mathbf{c}_5$ med 549 gener, fra Sørlie *et al.* (2003). For hver sentroide $\mathbf{c}_i$ ble det trukket tilfeldig 25 vektorer $\mathbf{x}_{ij}, j = 1, \ldots, 25$, fra normalfordelingen $N(\mathbf{c}_i, \sigma_1^2 I)$. I alt 50 slike datasett, hvert bestående av 125 vektorer, ble trukket. Dette ble så gjentatt for 18 ulike verdier av $\sigma_1$ mellom 1.0 og 2.7. Variansen er lik for alle fem klustrene, selv om dette er en forenkling i forhold til en virkelig situasjon. (Se avsnitt 6.2 for en situasjon med ulik varians for de fem klustrene.)

Et mål med denne simuleringen er å finne en grense for når gapobservatoren bryter sammen. Vi vil da finne den variansen hvor gapobservatoren ikke greier å skille klustrene i flertallet av simuleringene.

### 6.1.1 Resultater

Tabell 6.1 viser resultatet fra de $18 \cdot 50 = 900$ simuleringene. Her er kvadratroten av variansen gitt, altså standardavviket.

| $\sigma_1$ \ $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.0 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 1.1 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 1.2 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 1.3 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 1.4 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 1.5 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 1.6 | 0 | 0 | 0 | 3 | 47 | 0 | 0 | 0 | 0 | 0 |
| 1.7 | 0 | 0 | 3 | 4 | 43 | 0 | 0 | 0 | 0 | 0 |
| 1.8 | 0 | 0 | 8 | 6 | 36 | 0 | 0 | 0 | 0 | 0 |
| 1.9 | 0 | 0 | 8 | 6 | 36 | 0 | 0 | 0 | 0 | 0 |
| 2.0 | 0 | 3 | 22 | 8 | 17 | 0 | 0 | 0 | 0 | 0 |
| 2.1 | 0 | 8 | 26 | 9 | 7 | 0 | 0 | 0 | 0 | 0 |
| 2.2 | 0 | 20 | 22 | 6 | 2 | 0 | 0 | 0 | 0 | 0 |
| 2.3 | 0 | 38 | 10 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2.4 | 0 | 42 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2.5 | 0 | 46 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2.6 | 3 | 45 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2.7 | 8 | 41 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabell 6.1: Resultat-tabell fra kjøringer av GapObs. Her er $k$ antall foreslåtte klustre og $\sigma_1$ standardavviket rundt alle klustersentrene fra Sørlie *et al.* (2003).

Et godt resultat for gapobservatoren i dette tilfellet er å gi 5 klustre. Fra tabell 6.1 ser vi at dette er resultatet 50 av 50 ganger for standardavvik mellom 1.0 og 1.5. For større $\sigma_1$ begynner vi å se en tendens til at det velges færre klustre. Hvis vi øker $\sigma_1$ ytterligere, til rundt 2.0, ser vi at gapobservatoren bryter sammen. Hovedtyngden av resultatene ligger nå på $\hat{k} = 3$. De fleste kjøringer med dette standardavviket gir altså 3 klustre. Vi ser aldri at hovedtyngden ligger på 4 klustre, men går rett fra 5 til 3 klustre når vi øker standardavviket.

Videre ser vi at gapobservatoren gir 2 klustre fra de fleste av de 50 simuleringene, når standardavviket er økt til 2.3. Øker vi det til mer enn 2.5, vil flere og flere kjøringer gi ett kluster som resultat.

### 6.1.2 Diskusjon

Fra resultatene ovenfor kan vi se at gapobservatoren med Sørlie-sentroidene tåler et standardavvik opp mot 2.0 før metoden bryter sammen. Til sammenlikning forventes det gjennomsnittlige standardavviket for fem klustre fra hele intrinsic Sørlie-datasettet å ligge på godt over 2.0. (Dette støttes av tabell 6.2 på side 49 som viser standardavvikene fra dette datasettet, basert på de fem klustrene som Sørlie *et al.* (2003) har identifisert. Fra individene som er inneholdt i disse klustrene, som er ca 60% av individene, er det gjennomsnittlige standardavviket for de fem klustrene i underkant av 1.2.)

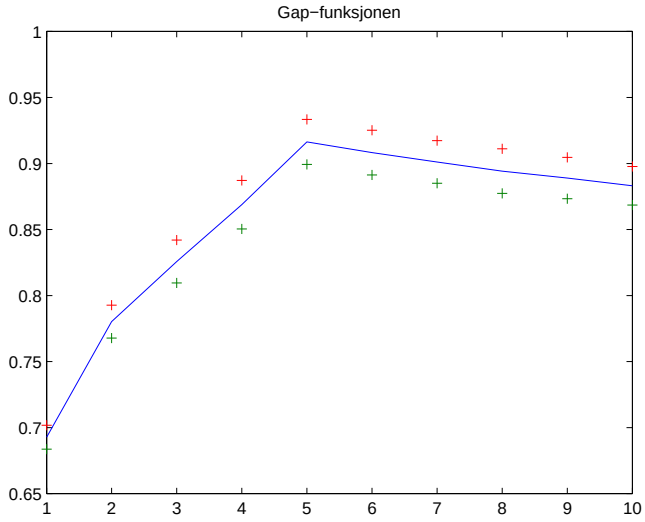
Når spredningen rundt sentroidene kommer opp på et nivå rundt 2.0, begynner klustrene i henhold til gapkriteriet å gli inn i hverandre. Punktene fra de opprinnelig fem klustrene har da begynt å nærme seg andre klustersentre enn de er simulert fra, og gapobservatoren klarer ikke lenger å skille de fem klustrene.

I kjøringene av GapObs vist i tabell 6.1 har vi tillatt opptil $K = 10$ klustre. Vi får imidlertid ikke et eneste resultat over fem klustre. Gapobservatorens kriterium for valg av antall klustre sier at vi skal velge det første antall klustre som gir en verdi større enn verdien for det neste antall klustre minus et standardavvik. Det vil si at resultatet favoriseres for det første antall klustre som tilfredsstiller kriteriet og det blir vanskeligere og vanskeligere å velge en større $k$. Dette blir vanskeligere fordi gap-kurven hele tiden må stige med en viss størrelse for ikke å bli fanget opp av kriteriet. Dette gir en begrensning for når første antall klustre vil bli valgt.

I eksemplene ovenfor ser vi at gapobservatoren aldri overestimerer antall klustre. Den gir i en del tilfeller for lave estimater, noe som støtter en ny tilpasning/versjon av GapObs (se kapittel 7 om en rekursiv versjon av gapobservatoren). Siden GapObs aldri gir et overestimat kan vi altså se på gapobservatoren som en konservativ estimator.

Figur 6.1 på neste side viser resultatet av gapobservatoren brukt på et simulert datasett der punktene er simulert med lav varians fra sentroidene. Den velger her fem klustre fordi referansedataene for hvert punkt på gap-kurven har lavt standardavvik i forhold til stigningen til neste punkt. Hvis et høyt antall klustre skal velges, er det derfor ikke nødvendig at kurven stiger like bratt som for en situasjon der referansedataene har et høyere standardavvik.

Videre kan vi se på den samme simuleringen når gapobservatoren skal analysere seks klustre. Vi må da dele opp et tett kluster, som altså har lav varians, og det vil ikke gi stor nok gevinst i innen-gruppe spredningen. Dette kan derimot




Figur 6.1: Gap-kurve fra gapobservatoren brukt på simulering rundt sentroider fra intrinsic Sørlie-datasettet med varians lik 1.0, $\hat{k} = 5$. Plusstegnene viser ett standardavvik over og under hvert punkt på kurven.

gjøre at kurven begynner å falle fordi referansedatasettet har større gevinst av å deles opp enn selve datasettet. Resultatet er da at differansen mellom de to datasettene blir mindre og kurven faller. Figur 6.1 viser et tydelig topp-punkt på gapkurven og derfor at simuleringer med de samme verdiene ikke vil gi andre resultater, hverken lavere eller høyere.

Simuleringene er basert på Sørlie-datasettet sentroider, og vi vil derfor forvente at resultatene har en relativt sterk relasjon spesielt til intrinsic Sørlie-datasettet. Spørsmålet blir da om grunnen til at gapobservatoren bare velger to klustre for intrinsic Sørlie-datasettet, er at de virkelige individene er såpass spredt rundt disse klustersentroidene at metoden ikke klarer å skille klustrene. I tillegg dukker det opp et spørsmål om det er reellt å la alle klustrene ha lik varians. Jeg vil som svar på dette se på en situasjon der hvert av klustrene er simulert med samme varians for alle gener, mens variansen i ulike klustre er forskjellig.

## 6.2 Simuleringer med ulik støy

Dette forsøket går ut på å simulere rundt klustersentroidene fra intrinsic Sørlie-datasettet, med ulik varians for hvert av de fem klustrene. De individene som har klustret sammen er spesifisert i *Supporting Table 3* (Sørlie *et al.,* 2003). Denne tabellen er sammenliknet med individene fra intrinsic Sørlie-datasettet, og individene er sortert i de samme fem klustrene som er publisert i Sørlie *et al.* (2003). Jeg satt da igjen med 79 individer fordelt på fem klustre, og 43 individer som ikke er med i noe kluster, dette var de samme tallene som i artikkelen til Sørlie *et al.* (2003).

Fra disse fem klustrene finner vi variansene til de respektive klustersentroidene nevnt i innledningen til kapittel 6. Antall individer og standardavvik for hvert kluster er gitt i tabell 6.2.

| Kluster: | Luminal A | Luminal B | Erbb2+ | Basal | Normal-like | Totalt |
|---|---|---|---|---|---|---|
| Antall ind. | 28 | 11 | 11 | 19 | 10 | 79 |
| Std. avvik: | 1.2103 | 1.0266 | 0.9920 | 1.4760 | 1.1203 | |

Tabell 6.2: Sammendrag av verdier for hvert kluster, basert på intrinsic Sørlie-datasettet og de fem definerte klustrene fra Sørlie *et al.* (2003).

Jeg har simulert 25 individer rundt hvert klustersenter med ulik varians for hvert av klustrene (som oppgitt i tabell 6.2). Dette ble gjort ved å variere gjennomsnittlig standardavvik for klustrene. Jeg har gjort 3 ulike simuleringer, der den første simuleringen bruker den opprinnelige gjennomsnittlige variansen for klustrene i tabell 6.2, mens de to neste simuleringene er gjort med gjennomsnittlig varians på $1.9^2 = 3.61$ og $2.1^2 = 4.41$. Det vil si i det området hvor vi ser at gapobservatoren bryter sammen. (Se tabell 6.1 på side 46.)

De nye standardavvikene er regnet ut ved at gjennomsnitts*variansen* for klustrene holdes fast, mens klustrenes egen varians økes og minskes. Det er innført en variabel $\beta \geq -1.0$ som angir i hvilken grad klustrenes egen varians skal avvike fra gjennomsnittet. Gitt variansene $\sigma_i^2$ for klustrene $i = 1, \ldots, 5$ er gjennomsnittsvariansen

$$\bar{\sigma}^2 = \frac{1}{5} \sum_{i=1}^{5} \sigma_i^2 \quad .$$

En ny varians finnes for hvert av klustrene

$$\tilde{\sigma}_i^2 = \sigma_i^2 + \beta(\sigma_i^2 - \bar{\sigma}^2) \quad .$$

Følgende vil være tilfelle for gitte verdier av $\beta$:

- For $\beta = -1.0$ vil $\tilde{\sigma}_i^2 = \bar{\sigma}^2, \quad \forall\ i$.
- For $\beta = 0.0$ vil $\tilde{\sigma}_i^2 = \sigma_i^2, \quad \forall\ i$.
- For $\beta > 0.0$ vil $\tilde{\sigma}_i^2 - \sigma_i^2 > 0, \quad \forall\ i$ og denne differansen vil øke med $\beta$.

  For klustre $\sigma_i^2 \neq \bar{\sigma}^2$ gjelder følgende:

  - For klustre med $\sigma_i^2 < \bar{\sigma}^2$ vil $\tilde{\sigma}_i^2 < \sigma_i^2$ og bli et tettere kluster.
  - For klustre med $\sigma_i^2 > \bar{\sigma}^2$ vil $\tilde{\sigma}_i^2 > \sigma_i^2$ og bli et bredere kluster.

### 6.2.1 Resultater

Jeg har tre resultattabeller. Den første tabellen (tabell 6.3(a) ) inneholder resultater fra simuleringer der gjennomsnittsvariansen er lik den opprinnelige, mens de to neste tabellene (tabell 6.3(b) og tabell 6.3(c) ) er resultater fra simuleringer med en gjennomsnittsvarians på henholdsvis $1.9^2 = 3.61$ og $2.1^2 = 4.41$.

Tabell 6.3: De tre tabellene viser resultater fra GapObs og standardavvik for hver verdi av $\beta$. De gjennomsnittlige variansene for de tre tabellene er henholdsvis lik gjennomsnittsvariansen fra tabell 6.2 som er $1.1778^2 = 1.3872$, $1.9^2 = 3.61$ og $2.1^2 = 4.41$.

| $\beta$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| -1.0 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 0.0 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 1.0 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 2.0 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |

| $\beta$ | Luminal A | Luminal B | Erbb2+ | Basal | Normal-like |
|---|---|---|---|---|---|
| -1.0 | 1.1778 | 1.1778 | 1.1778 | 1.1778 | 1.1778 |
| 0.0 | 1.2103 | 1.0266 | 0.9920 | 1.4760 | 1.1203 |
| 1.0 | 1.2420 | 0.8489 | 0.7622 | 1.7233 | 1.0596 |
| 2.0 | 1.2728 | 0.6224 | 0.4216 | 1.9393 | 0.9953 |

(a) Øverst er det gitt resultater fra GapObs. Det er gjort 50 simuleringer for hver verdi av $\beta$, gjennomsnittet av variansen er holdt fast lik gjennomsnittsvariansen fra tabell 6.2, $1.1778^2 = 1.3872$. Nederst er det gitt standardavvikene for hver verdi av $\beta$.

| $\beta$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| -1.0 | 0 | 0 | 8 | 4 | 38 | 0 | 0 | 0 | 0 | 0 |
| 0.0 | 0 | 0 | 0 | 11 | 39 | 0 | 0 | 0 | 0 | 0 |
| 1.0 | 0 | 0 | 0 | 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 2.0 | 0 | 1 | 5 | 15 | 29 | 0 | 0 | 0 | 0 | 0 |

| $\beta$ | Luminal A | Luminal B | Erbb2+ | Basal | Normal-like |
|---|---|---|---|---|---|
| -1.0 | 1.9000 | 1.9000 | 1.9000 | 1.9000 | 1.9000 |
| 0.0 | 1.9524 | 1.6561 | 1.6003 | 2.3809 | 1.8072 |
| 1.0 | 2.0034 | 1.3694 | 1.2295 | 2.7798 | 1.7093 |
| 2.0 | 2.0532 | 1.0040 | 0.6801 | 3.1283 | 1.6055 |

(b) Øverst er det gitt resultater fra GapObs. Det er gjort 50 simuleringer for hver verdi av $\beta$, gjennomsnittet av variansen er holdt fast lik $1.9^2 = 3.61$. Nederst er det gitt standardavvikene for hver verdi av $\beta$.

| $\beta$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| -1.0 | 0 | 9 | 25 | 10 | 6 | 0 | 0 | 0 | 0 | 0 |
| 0.0 | 0 | 6 | 23 | 17 | 4 | 0 | 0 | 0 | 0 | 0 |
| 1.0 | 0 | 1 | 1 | 12 | 36 | 0 | 0 | 0 | 0 | 0 |
| 2.0 | 0 | 2 | 18 | 15 | 15 | 0 | 0 | 0 | 0 | 0 |

| $\beta$ | Luminal A | Luminal B | Erbb2+ | Basal | Normal-like |
|---|---|---|---|---|---|
| -1.0 | 2.1000 | 2.1000 | 2.1000 | 2.1000 | 2.1000 |
| 0.0 | 2.1579 | 1.8304 | 1.7687 | 2.6315 | 1.9974 |
| 1.0 | 2.2143 | 1.5136 | 1.3589 | 3.0724 | 1.8893 |
| 2.0 | 2.2693 | 1.1097 | 0.7517 | 3.4576 | 1.7745 |

(c) Øverst er det gitt resultater fra GapObs. Det er gjort 50 simuleringer for hver verdi av $\beta$, gjennomsnittet av variansen er holdt fast lik $2.1^2 = 4.41$. Standardavvikene for hver verdi av $\beta$ er også gitt.

### 6.2.2 Diskusjon

Resultatene for $\beta = -1.0$ kan sammenliknes direkte med resultatene fra tabell 6.1 på side 46. For $\beta = -1.0$ har alle simulerte fordelinger (alle klustre) likt standardavvik.

I tabell 6.3(a) ser vi på resultater fra simuleringer med standardavvik mindre enn 2.0, som var den verdien vi observerte at gapobservatoren bryter sammen (se tabell 6.1). Vi ser at kun standardavviket for kluster 4, *Basal*-klusteret, nærmer seg en verdi der vi kan forvente at gapobservatoren ikke klarer å skille dette klusteret fra de andre.

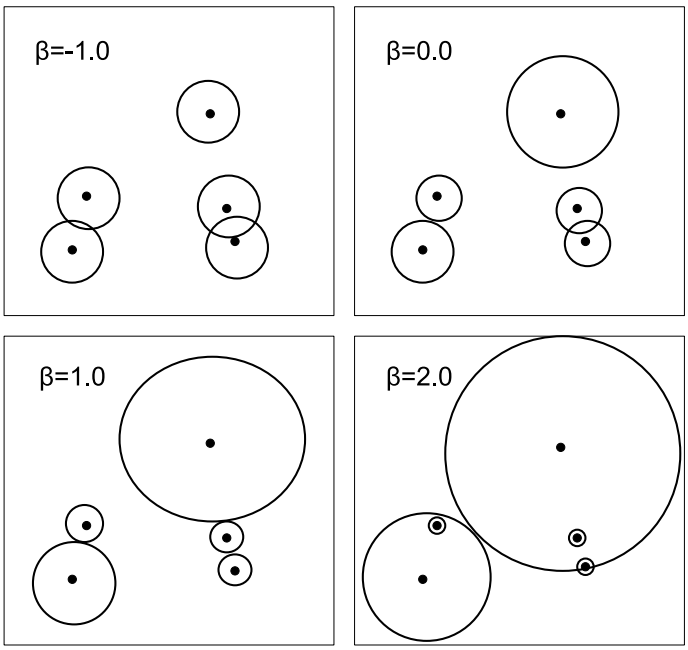
Utgangspunktet for simuleringen var klustersentre gitt fra Sørlie *et al.* (2003). Disse klustersentrene er ikke beregnet fra alle individene i intrinsic Sørlie-datasettet, men kun fra noen utvalgte individer. (Individene er gitt i Supporting Table 3, fra Sørlie *et al.* (2003).) Fra de individene som ble brukt for å beregne klustersentrene, har jeg regnet ut standardavviket for hvert kluster. Resultatet var derfor forventet å gi $\hat{k} = 5$ for alle simuleringer. Dette viste seg å være tilfelle, også når vi forandret $\beta$. Vi ser fra tabellen at alle standardavvik er under 2.0, noe som understøtter det at gapobservatoren klarer å skille klustrene når standardavviket ikke overstiger 2.0.

Når vi videre ser på resultatene fra de neste simuleringene, gitt i tabellene 6.3(b) og 6.3(c), kunne vi i utgangspunktet forvente en monoton utvikling når vi øker $\beta$. Dette observerer vi allikevel ikke. Utviklingen går mot et mer presist resultat på $\hat{k} = 5$ for $\beta = 1.0$ og blir så mindre presist igjen for større $\beta$.

Det er vanskelig å kunne visualisere et slikt tilfelle, fordi individene i klustrene ligger i et høydimensjonalt rom, i dette tilfellet i over 500 dimensjoner. Siden det er vanskelig å visualisere noe i så høy dimensjon, så jeg vil prøve å se på denne situasjonen som punkter i et to dimensjonalt rom.

En mulig forklaring på den ikke-monotone utviklingen kan være at klusteresentrene ligger i en bestemt avstand i det $p$-dimensjonale rommet, som tilsier at resultatet blir som vi observerer i tabellene 6.3(b) og 6.3(c). Et eksempel på dette er vist i figur 6.2 på neste side. Dette er et to-dimensjonalt eksempel på en utvikling som kan forklare den ikke-monotone utviklingen. Tabell 6.3(a) er basert på varianser av så liten størrelse at forandring av $\beta$ ikke utgjør noen merkbar forskjell i resultatet fra gapobservatoren.

Tabell 6.3(b) er derimot et godt eksempel på denne ikke-monotone utviklingen. Ved å ta utgangspunkt i figurens rute for $\beta = 0.0$, der variansene har den opprinnelige spredningen om varians-gjennomsnittet, ser vi at overlappet mel-




Figur 6.2: Figuren illustrerer et tilfelle som i tabell 6.3(b) og viser fem klustersentre som prikker og variansene om klustersentrene som sirkler, for gitt verdi av $\beta$. Øverst t.v: $\beta = -1.0$. Alle klustre har lik varians og noen klustre overlapper. Gapobservatoren vil kunne velge 3 klustre. Øverst t.h: $\beta = 0.0$. Noen av klustrene har større og noen har mindre varians enn gjennomsnittet. Gapobservatoren vil trolig velge 4 klustre. Nederst t.v: $\beta = 1.0$. Variansene er mer forskjøvet fra gjennomsnittet og vi ser ingen overlapp mellom klustrene. Gapobservatoren vil derfor trolig velge 5 klustre. Nederst t.h: $\beta = 2.0$. Variansene er enda mer spredt og klustre vil igjen overlappe. Gapobservatoren vil her i det minste gi 2 klustre.

lom klustrene er lite. Figuren viser at resultatet kan gi 4 eller 5 klustre, som også er tilfellet i tabell 6.3(b). Ser vi videre på figur 6.2 sin rute for $\beta = -1.0$, ser vi at alle variansene er like store. Dette gjør at flere klustre overlapper, noe som kan gi 3-5 klustre, men med såpass lite overlapp at det å velge 5 klustre vil være mest naturlig.

Går vi videre til figur 6.2 sin rute for $\beta = 1.0$, ser vi at de klustrene med størst varians har fått større varians enn for $\beta = 0.0$, og motsatt har klustrene med minst varians fått mindre varians enn for $\beta = 0.0$. I tilfellet som er skissert i figuren ser vi at klustrene har blitt helt adskilt og gapobservatoren vil gi 5 klustre. Når vi videre øker $\beta$ til 2.0, har forskjellene mellom klustervariansene blitt enda større og vi har igjen fått overlapp mellom klustrene. Det er nå klustrene med størst varians som har fått så stor varians at disse klustrenes punkter overlapper

de tette klustrene. Dette tilfellet viser at gapobservatoren kan gi fra 2-5 klustre, noe som også er tilfellet i tabell 6.3(b).

Figur 6.2 er kun en enkel illustrasjon og det er ikke tydeliggjort i figuren at variansen holdes konstant, selv om dette er tilfelle i simuleringene. Illustrasjonen er allikevel vist for å antyde at det er mulig å finne visuelle eksempler på at utviklingen i tabell 6.3(b) er reell.

Eksempelet i figur 6.2 viser altså at den ikke-monotone uviklingen er mulig å forklare. Vi vet at de simulerte datasettene er basert på 5 klustre, men når vi bruker gapobservatoren på et virkelig datasett vet vi ikke hvor mange klustre som er optimalt. Vi vet derfor ikke om de virkelige data er i en situasjon som er lik den for $\beta = -1.0$ eller nærmere $\beta = 2.0$. Vi vet altså ikke om virkelige klustre overlapper og glir inn i hverandre. Det vi derfor spør oss, er om gapobservatoren gir et lavere resultat enn det som er riktig for datasettet eller om resultatet er det optimale?

Det kan altså være vanskelig å avgjøre om resultatet er optimalt for et virkelig datasett. Én ting er vi derimot mer sikre på, som vi kan se fra resultatene av simuleringene, og det er at vi ikke får et resultat som er større enn det som er riktig for datasettet. Vi får enten et lavere resultat eller et resultat som er likt det optimale. Det vil si at vi enten har et overlapp mellom klustre som gapobservatoren ikke greier å skille eller vi har funnet riktig antall klustre.

Et forslag til hvordan man kan gå frem videre er da å bruke en metode som ser nærmere på de klustrene vi har. Fra denne metoden får vi da enten en bekreftelse på at vi har nådd det optimale eller vi får et hint om at ett eller flere av klustrene fortsatt kan deles opp i flere klustre. En metode som gjør dette er foreslått i kapittel 7.

## 6.3 Simulering med uteliggere

I en reell situasjon er det normalt at datasettet inneholder uteliggere. Jeg vil derfor også presentere en situasjon der jeg simulerer et datasett med uteliggere. Jeg vil gjøre dette ved at en gitt prosentandel av målingene har en større varians enn resten av målingene.

Simuleringene er gjort som før, rundt de fem sentroidene fra Sørlie *et al.* (2003). Fra hver av disse sentroidene har jeg simulert 25 individer, hvor en andel $\alpha$ av verdiene (målingene for hver probe for hvert individ) er tillagt ekstra støy. Disse verdiene simulerer uteliggere og er simulert fra en normalfordeling med

standardavvik som er 3 ganger større enn det som er gitt for de andre verdiene. Standardavviket er likt for alle klustrene.

Verdiene er simulert med konstant samlet varians i to ulike situasjoner med $\sigma_1 = 1.8$ og $\sigma_1 = 1.9$. Dette har vist seg å være passende verdier, fordi vi da er nær grensen hvor gapobservatoren bryter sammen. Jeg har holdt den samlede variansen konstant ved å gjøre følgende.

- Jeg har trukket individer fra to normalfordelinger med ulik varians og samme gjennomsnitt. Andelen $(1 - \alpha)$ er individer som skal trekkes fra en normalfordeling med varians $\sigma_1^2$ og andelen $\alpha$ er individer som trekkes fra en normalfordeling med varians $\sigma_2^2$.
- Summen av den samlede variansen er da $\sigma_{\text{tot}}^2 = (1 - \alpha)\sigma_1^2 + \alpha\,\sigma_2^2$.
- Datasettet er nå gitt som $X = (x_{ij})$, der $i = 1, \ldots, n$ individer og $j = 1, \ldots, p$ gener. For å holde den samlede variansen konstant har jeg standardisert datasattet:

$$\tilde{x}_{ij} = x_{ij}/\sigma_{\text{tot}} \qquad \forall i, j.$$

### 6.3.1 Resultater

Resultatene i tabell 6.4 og 6.5 på neste side er gitt for økende verdi av $\alpha$, som altså er andelen uteliggere i simuleringen.

| $\alpha$ \ $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.00 | 0 | 0 | 0 | 4 | 46 | 0 | 0 | 0 | 0 | 0 |
| 0.01 | 0 | 0 | 0 | 6 | 44 | 0 | 0 | 0 | 0 | 0 |
| 0.02 | 0 | 0 | 0 | 3 | 47 | 0 | 0 | 0 | 0 | 0 |
| 0.03 | 0 | 1 | 1 | 8 | 40 | 0 | 0 | 0 | 0 | 0 |
| 0.04 | 0 | 0 | 0 | 9 | 41 | 0 | 0 | 0 | 0 | 0 |
| 0.05 | 0 | 0 | 0 | 6 | 44 | 0 | 0 | 0 | 0 | 0 |
| 0.06 | 0 | 0 | 0 | 6 | 44 | 0 | 0 | 0 | 0 | 0 |
| 0.07 | 0 | 0 | 0 | 11 | 39 | 0 | 0 | 0 | 0 | 0 |
| 0.08 | 0 | 0 | 0 | 7 | 43 | 0 | 0 | 0 | 0 | 0 |
| 0.09 | 0 | 0 | 1 | 13 | 36 | 0 | 0 | 0 | 0 | 0 |
| 0.10 | 0 | 0 | 0 | 11 | 39 | 0 | 0 | 0 | 0 | 0 |

Tabell 6.4: Resultat-tabell fra kjøringer av GapObs på datasett med uteliggere og varians $\sigma_1 = 1.8$ og $\sigma_2 = 3\sigma_1$. Her er $k$ antall foreslåtte klustre og $\alpha$ andelen uteliggere.

| $\alpha$ \ $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.00 | 0 | 0 | 0 | 5 | 45 | 0 | 0 | 0 | 0 | 0 |
| 0.01 | 0 | 0 | 0 | 14 | 36 | 0 | 0 | 0 | 0 | 0 |
| 0.02 | 0 | 0 | 1 | 6 | 43 | 0 | 0 | 0 | 0 | 0 |
| 0.03 | 0 | 0 | 4 | 11 | 35 | 0 | 0 | 0 | 0 | 0 |
| 0.04 | 0 | 0 | 0 | 11 | 39 | 0 | 0 | 0 | 0 | 0 |
| 0.05 | 0 | 0 | 2 | 6 | 42 | 0 | 0 | 0 | 0 | 0 |
| 0.06 | 0 | 0 | 0 | 12 | 38 | 0 | 0 | 0 | 0 | 0 |
| 0.07 | 0 | 0 | 3 | 17 | 30 | 0 | 0 | 0 | 0 | 0 |
| 0.08 | 0 | 0 | 0 | 9 | 41 | 0 | 0 | 0 | 0 | 0 |
| 0.09 | 0 | 0 | 0 | 12 | 38 | 0 | 0 | 0 | 0 | 0 |
| 0.10 | 0 | 0 | 0 | 19 | 31 | 0 | 0 | 0 | 0 | 0 |

Tabell 6.5: Resultat-tabell fra kjøringer av GapObs på datasett med uteliggere og varians $\sigma_1 = 1.9$ og $\sigma_2 = 3\sigma_1$. Her er $k$ antall foreslåtte klustre og $\alpha$ andelen uteliggere.

### 6.3.2 Diskusjon

I dette tilfellet er data simulert fra 5 klustersentre og det optimale antall klustre er derfor 5 klustre. Tabellene 6.4 og 6.5 viser utviklingen etter hvert som andelen uteliggere økes. Fra disse resultatene ser vi at antallet kjøringer av GapObs som gir 5 klustre synker når vi øker andelen uteliggere. Vi ser derfor at antallet kjøringer av GapObs som gir færre enn 5 klustre øker når andelen uteliggere øker.

De to tabellene har som sagt forskjellig varians. Tabell 6.4 har $\sigma_1 = 1.8$, mens tabell 6.5 har $\sigma_1 = 1.9$. Det forventes derfor at tabell 6.4 viser et større antall kjøringer som resulterer i 5 klustre enn tabell 6.5, fordi tabell 6.5 viser resultater fra GapObs på datasett med større spredning. Vi oberserverer at dette er tilfelle.

Resultatene viser en svak tendens til at gapobservatoren velger færre klustre for økende andel uteliggere. Det vil si at stabiliteten til gapobservatoren påvirkes av andelen uteliggere, noe vi i utgangspunktet forventer.

Når andelen uteliggere blir stor vil skillet mellom klustrene kunne bli mindre i noen dimensjoner fordi uteliggerne er tillagt ekstra støy i disse dimensjonene. Det vil si at punkter som har ekstra støy lagt til vil i noen dimensjoner kunne bli liggende mellom klustre og viske ut skillet mellom klustrene. Dette vil resultere i at gapobservatoren kan velge et mindre antall klustre. En slik utvikling vil vi forvente, men vi kan se fra resultat-tabellene at denne tendensen faktisk er svak.

Vi ser også at det sjelden velges færre enn 4 klustre. Dette kan forklares med

at den ekstra støyen som blir lagt til disse uteliggerne, kun gjelder enkelte dimensjoner. Simuleringen utføres som beskrevet tidligere slik at det trekkes tilfeldige verdier som får tillagt ekstra støy og derfor ikke slik at alle verdiene til utvalgte punkter (individer) blir simulert som uteliggere. Siden den ekstra støyen blir lagt til punktene i ulike dimensjoner, vil ikke slik støy alltid påvirke resultatet.

# Kapittel 7

# Rekursiv versjon av gapobservator-metoden

Vi har tidligere sett på en versjon av gapobservatoren som gir et resultat i form av antall klustre som sees på som det optimale i følge gapobservatoren. En rekursiv versjon av gapobservatoren er en algoritme som først kjører GapObs. For hvert av de resulterende klustrene kjøres så GapObs rekursivt. Rekursiviteten stopper når et kall på GapObs resulterer i *ett* kluster. Det vil si at denne versjonen estimerer antall klustre først i datasettet og estimerer så antall klustre i de resulterende klustrende. Slik fortsetter metoden å estimere antall klustre i resultatklustrene, helt til antallet klustre som estimeres i hvert av de resulterende klustrene er én.

Dette betyr at metoden vil gi minst samme antall klustre som GapObs. Jeg har kalt implementasjonen av denne metoden for RecGapObs.

## 7.1 Beskrivelse av den rekursive gapobservator-metoden

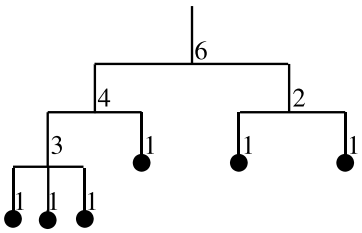
RecGapObs forløper som beskrevet i algoritme 1 på neste side. (Se avsnitt 10.2 på side 73 for implementasjon i Matlab.) Metoden forventer fire parametre:

- *data*: et datasett der radene vil bli klustret
- *K*: maksimalt antall klustre som vil bli testet
- *B*: antall simuleringer av referansedatasett
- *names*: individ-id'er, kan være tom ([]).

**Algoritme 1** Forløp av RecGapObs

- Kjør GapObs med gitte parametre. Dette vil gi et estimert antall klustre $\tilde{k}_i$, for denne $i$'te kjøringen av GapObs.
- For hvert av de $\tilde{k}_i$ klustrene:
  - Hvis $\tilde{k}_i = 1$, gå ut av løkka
  - Ellers, kjør RecGapObs rekursivt
- Hvis $\tilde{k}_i = 1$, returnér verdien 1.
- Ellers, summér returverdiene fra kjøringene av RecGapObs og returnér denne summen.

Figur 7.1 viser et rekursjonstre for RecGapObs og hvilke verdier (antall klustre) som returneres. Det returneres ingenting før resultatet er 1 kluster. Summeringen starter derfor i bladnodene i dette treet og summerers oppover i treet. I dette tilfellet vil RecGapObs foreslå 6 klustre.




Figur 7.1: Eksempel på et rekursjonstre for RecGapObs. Bladnodene er et kluster som ikke deles opp av GapObs. Tallene er returverdier fra hver kjøring av RecGapObs.

Utgangspunktet til gapobservatoren er et $n \times p$ datasett $X$, med $n$ individer og $p$ gener. Vi ser da på $n$ individer i et $p$-dimensjonalt rom og gapobservatoren finner ifølge sitt kriterium, et optimalt antall klustre. Det bygges da først opp et referansedatasett rundt prinsipalretningene til datasettet (som beskrevet i avsnitt 5.1 på side 33). Videre beregnes innen-gruppe avstander mellom individene i datasettet og $1, ..., K$ klustersentre, og innen-gruppe avstander mellom referanseindivider og $1, ..., K$ nye klustersentre.

Gapobservatoren sammenlikner altså datasettet med et referansedatasett, der begge datasett ligger i samme prinsipalretninger og innenfor samme retninger.

Vi kan da se for oss individene som punkter i en $p$-dimensjonal boks som er lagt rundt datasettet vårt og trukket nye individer til et referansedatasett fra samme boksen. Hvis vi da antar at datasettet vårt har en viss struktur og at individene grupperer seg naturlig i mer enn ett kluster, så vil vi kunne se at en veldig stor del av volumet av boksen ikke har noen individer i nærheten. Dette gjør at gapobservatoren leter etter klustre i et veldig stort rom uten individer, og må se på individene fra et ståsted langt unna boksen. Det blir som å se på et bilde uten forstørrelse av det vi ønsker å fokusere på. Det er derfor vanskelig å tydelig kunne skille ut klustre.

Hvis vi videre ber gapobservatoren kun lete etter klustre i de områdene det befinner seg individer, vil vi kunne se strukturen av fordelingen av individer fra et nærmere ståsted. Dette blir som å se på individene med et forstørrelsesglass, og se om det da er mulig å skille individene ved hjelp av det samme kriteriet. Kriteriet (5.2) brukes som før, men sammenlikningsgrunnlaget består av en tettere samling med referanseindivider. Det er nettopp dette som er kjernen i algoritmen RecGapObs.

I dimensjoner av den størrelsen som ofte sees i et genekspresjonsdatasett, vil én kjøring av GapObs trolig ofte gi et lavere resultat enn det som er optimalt for datasettet. Et slikt resultat får vi nettopp fordi vi ser på et så høydimensjonalt rom at det skal mye til å velge et høyere antall klustre. Dette skyldes at gevinsten ved å dele datasettet i kun to klustre ofte er så stor at kriteriet blir oppnådd allerede da. Det vil si at selv om kriteriet blir oppnådd for et høyere antall klustre, så blir det første antall klustre som tilfredstiller kriteriet valgt. Dette gjør at vi vil undersøke hvert kluster mer nøyaktig, for å sjekke om det finnes en naturlig oppdeling av hvert kluster i flere subklustre.

Hvert rekursivt kall på metoden RecGapObs analyserer altså en delmengde av datasettet, ved at metoden ser på den betingete sannsynligheten for at et individ fra et gitt resultatkluster ligger i et gitt subkluster. Betingelsen er da at individet må klustres til et subkluster av det opprinnelige klusteret, og ikke kan inngå i et subkluster av et annet kluster. Dette bygger opp en hierarkisk struktur som stopper opp når det ikke er naturlig, ifølge gapobservatoren, å dele opp et subsett av datasettet i flere enn ett kluster.

## 7.2 Bruk av metoden på Sørlie-datasettet

### 7.2.1 Resultater

Jeg har implementert RecGapObs i Matlab (se kode i avsnitt 10.2 på side 73) og testet implementasjonen på intrinsic Sørlie-datasettet. Jeg har kjørt metoden 50 ganger med samme data, antall simuleringer $B = 100$, tillatt opptil $K = 10$ klustre og gitt ved individ-id'er for alle individer.

Resultatet fra dette er vist i tabell 7.1. Variasjonen her skyldes variasjonen i referansedata i gapobservator-metoden. Ved å øke antall simuleringte referansesett, ville vi få stadig mindre variasjon i resultatene.

| k: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Antall Klustre: | 0 | 0 | 0 | 0 | 48 | 2 | 0 | 0 | 0 | 0 |

Tabell 7.1: Resultat fra 50 kjøringer av RecGapObs hvor vi har tillatt opptil $K = 10$ klustre og med $B = 100$ simuleringer for hver kjøringer.

### 7.2.2 Diskusjon

Jeg har gjennom hele oppgaven sett på resultatene fra artikkelen til Sørlie *et al.* (2003) og de klustrene som er funnet der, som en fasit for intrinsic Sørlie-datasettet. I artikkelen har de argumentert for at det er biologisk grunnlag for å velge de fem klustrene *Luminal A*, *Luminal B*, *Erbb2+*, *Basal* og *Normal*. I tillegg har de vist at en hierarkisk klustring grupperer individene til de samme klustrene. For å oppnå dette resultatet er det blitt benyttet en del heuristikker. Blant annet er det luket bort individer som ikke korrelerer godt med klustersentrene. Allikevel ser jeg på artikkelen som en referanse for at det optimale antallet klustre i intrinsic Sørlie-datasettet er 5.

Det er derfor interessant at resultatet jeg får fra RecGapObs har hovedtyngde på nettopp 5 klustre. I 48 av 50 tilfeller gir RecGapObs $\hat{k} = 5$ som det optimale antall klustre, og i 2 av 50 tilfeller er resultatet $\hat{k} = 6$. Metoden foreslår altså alltid at antallet klustre for intrinsic Sørlie-datasettet er 5 eller mer.

Det er naturlig å spørre seg om individer som er klustret sammen i Sørlie *et al.* (2003), også klustrer sammen etter å ha kjørt RecGapObs. For å svare på dette kan vi sammenlikne klustringsresultatet fra artikkelen med resultatet fra RecGapObs. Sørlie *et al.* (2003) har brukt 79 individer i presentasjonen av klustringsresultatet. Da er altså de 43 individene med lav korrelasjon til sitt klustersenter fjernet. Jeg

har brukt alle 122 individene for å finne de fem klustrene. I tabell 7.2 og 7.3 har jeg sammenliknet klustringsresultatene og individene fra Sørlie *et al* og fra kjøringer av RecGapObs. Navngivningen i tabell 7.3 er gitt etter klustrene i Sørlie *et al.* og klustrene fra RecGapObs er nummererte og ordnet etter disse klustrene.

| | RecGapObs klustre | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| Luminal A | 21 | 0 | 0 | 0 | 7 |
| Luminal B | 0 | 11 | 0 | 0 | 0 |
| Errb2 | 0 | 0 | 11 | 0 | 0 |
| Basal | 0 | 0 | 0 | 19 | 0 |
| Normal | 1 | 0 | 0 | 0 | 9 |

Tabell 7.2: Individenes tilhørighet i klustrene gitt i Sørlie *et al.* (2003) og klustrene fra en kjøring av RecGapObs. Sistnevnte klustre er ordnet og nummerert for å lette tolkningen av resultatene, ved at de fleste observasjoner havner på diagonalen.

| | Luminal A | Luminal B | Erbb2+ | Basal | Normal-like | Totalt |
|---|---|---|---|---|---|---|
| 1) | 21 | 11 | 11 | 19 | 9 | 71 |
| 2) | 1 | 0 | 0 | 0 | 7 | 8 |
| 3) | 41 | 16 | 22 | 20 | 23 | 122 |

Forklaringer

| | |
|---|---|
| 1) | Antall individer som befinner seg i samme resultatkluster fra RecGapObs som i Sørlie *et al.* (2003). |
| 2) | Antall individer som er ulikt klustret i RecGapObs og i Sørlie *et al.* (2003). |
| 3) | Totalt antall individer i klustrene fra RecGapObs. |

Tabell 7.3: Sammenlikning av resultater fra Sørlie *et al.* (2003) og fra kjøringer av RecGapObs.

Fra tabell 7.2 og rad 2 i tabell 7.3, ser vi at RecGapObs har klustret veldig likt som Sørlie-klustrene. Det er kun 8 individer som er klustret forskjellig. I tillegg er det verdt å legge merke til at selv om det er brukt 43 flere individer i kjøringen av RecGapObs , så gir denne metoden essensielt samme resultat som i Sørlie *et al.* (2003). Det kan tenkes vi burde hatt 6 grupper av individer, der en av gruppene kun er for uteliggere. Uteliggere er da individer som ikke passer godt inn i de fem klustrene.

RecGapObs hevder altså at det er 5 klustre i intrinsic Sørlie-datasettet. Dendrogrammet i figur 4.2 på side 32 viser en initielle klustring av dette datasettet. Hvis vi fra denne figuren skal velge 5 klustre, ved å kutte på et gitt nivå i dendrogrammet, vil vi ende opp et kluster som inneholder kun 3 individer. Vi ser at RecGapObs i den samme situasjonen har 16 individer i det minste klusteret. Dette stemmer også bedre overens med den biologiske bakgrunnen i datasettet.

Det er faktisk en biologisk overenstemmelse med resultatet fra RecGapObs og intrinsic Sørlie-datasettet. Metoden viser seg altså å være nyttig for å predikere antall klustre i et slikt datasett. Jeg vil i kapittel 8 på neste side teste metoden på simulerte data, der jeg på forhånd vet hvilken relasjon de simulerte individene har til hverandre.

# Kapittel 8

# Testing av den rekursive gapobservator-metoden

Vi vet nå at RecGapObs gir et bedre resultat på intrinsic Sørlie-datasettet enn GapObs , gitt at Sørlie *et al.* (2003) har rett i at 5 klustre er det optimale. Jeg vil derfor gå videre og teste RecGapObs på simulerte datasett, for å få et bilde av hvordan metoden reagerer på kontrollerte datasett.

## 8.1 Simuleringer

Jeg vil simulere data fra 3 hovedklustre, som ligger tydelig adskilt. Videre vil jeg simulere 2 klustre fra hvert av hovedklustrene og til slutt trekke 20 punkter (individer) fra hver av disse 6 klustrene. Datasettet vil da være hierarkisk oppdelt og bestå av 120 simulerte individer.

Dette gjør jeg ved å la de 3 hovedklustersentrene være hjørner i en p-dimensjonal likesidet trekant. Jeg trekker så 2 nye klustersentre normaltfordelt rundt hver av de 3 hovedklustrene. Jeg har da 6 klustersentre som skal være utgangspunkt for individsimuleringen. Individene simuleres normalfordelt fra 6 normalfordelinger rundt de 6 klustersentrene.

Simuleringen av datasettene gjør jeg på tre ulike måter og får da følgende tre situasjoner:

1. Trekk 1 datasett som brukes i 100 kjøringer av GapObs og 100 kjøringer av RecGapObs. Denne situasjonen studerer variasjon i referansedatasettene i GapObs.

2. Trekk 100 ulike datasett basert på 1 trekning av klustersentre, men 100 ulike trekninger av individer, og bruk hvert datasett i én kjøring av GapObs og én kjøring av RecGapObs. Denne situasjonen studerer dermed individvariasjon.

3. Trekk 100 ulike datasett basert på 100 ulike trekninger av klustersentre og individer, og bruk hvert datasett i én kjøring av GapObs og én kjøring av RecGapObs. Denne situasjonen studerer variasjon i klustersentre og individer.

### 8.1.1 Resultater

Resultater fra de tre situasjonene gitt i forrige avsnitt:

1. Tabell 8.1(a) på side 67 viser 100 resultater fra kjøringer av GapObs og RecGapObs på et datasett simulert som beskrevet i punkt 1 over. Figur 8.1 på neste side viser en hierarkisk klustring av dette datasettet.

2. Tabell 8.1(b) på side 67 viser 100 resultater fra kjøringer av GapObs og RecGapObs på datasett simulert som beskrevet i punkt 2 over.

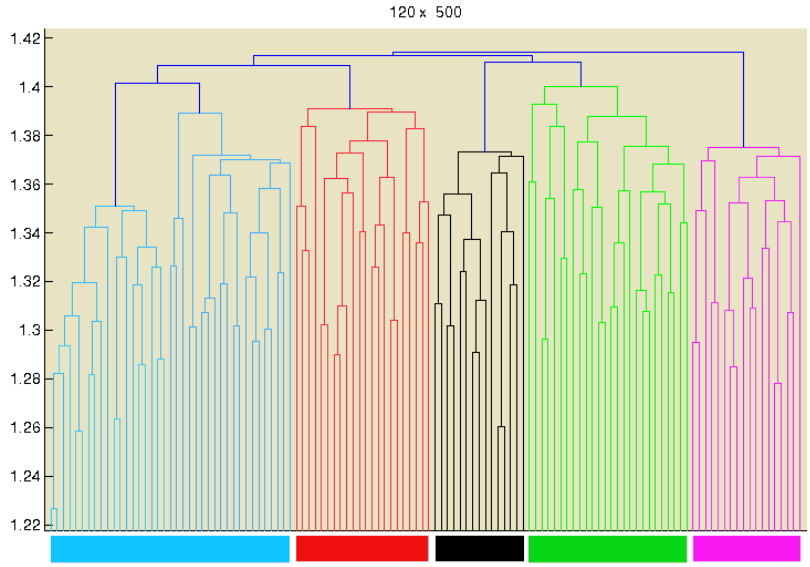
3. Tabell 8.1(c) på side 67 viser 100 resultater fra kjøringer av GapObs og RecGapObs på datasett simulert som beskrevet i punkt 3 over.

### 8.1.2 Diskusjon

Resultatet fra tabell 8.1(a) viser at GapObs foreslår et mindre antall klustre enn RecGapObs. I 90% av GapObs kjøringene velges det 3 klustre, mens 85% av RecGapObs tilfellene velger 5 klustre som det optimale. I dette tilfellet vet vi at datasettet er simulert fra 6 klustre og derfor er ingen av disse resultatene for store.

RecGapObs gir i tabell 8.1(a) et antall klustre som er nærmere det forventede enn det GapObs gir. Vi nærmer oss derfor det resultatet vi ønsker, nemlig 6 klustre, med et gapobservator-kriterium som bygger på det opprinnelige kriteriet (gitt i uttrykket (5.2) på side 36). Siden vi ikke har forandret dette kriteriet er argumentene for å velge nettopp dette kriteriet de samme som i artikkelen til Tibshirani *et al.* (2001).

I den 2. simuleringssituasjonen gir vi tilfeldigheten mer spillerom ved å simulere nye individer for hvert av de 100 datasettene. Den 3. simuleringssituasjonen gir oss enda mindre kontroll på resultatet, fordi vi i denne situasjon simulerer 6




Figur 8.1: Dendrogram som viser hierarkisk klustring av det simulerte datasettet brukt i situasjon 1. Fargekodene viser de 5 klustrene som RecGapObs valgte i 85 av 100 tilfeller. Ved å skille de to klustrene farget i blått vises de 6 klustrene som RecGapObs valgte i 13 av 100 tilfeller.

nye klustersentre og 20 nye individer for hvert nytt datasett. Vi har da kontroll på at de tre hovedklustrene ligger med lik avstand mellom seg, mens klustersentrene som er utgangpunktet for selve individsimuleringene trekkes normalfordelt for hvert datasett. Det gjør at vi i situasjonen 3 har størst tilfeldighet i simuleringen.

Vi vil på forhånd forvente en bredere fordeling av resultatene i situasjon 2 og 3 enn i situasjon 1. Dette er også tilfellet når vi sammenlikner resultatene fra tabell 8.1(b) og 8.1(c) med resultatene fra tabell 8.1(a).

Resultatet i tabell 8.1(b) viser en svak tendens til at både GapObs og RecGapObs velger flere klustre enn vist i tabell 8.1(c) ved å se på medianverdiene. Fordelingene i de to tabellene er ellers veldig like.

Når vi nå har testet GapObs og RecGapObs på flere datasett, ser vi at sistnevnte metode har en høyere medianverdi for resultatene. Det viser at RecGapObs tenderer til å velge et høyere antall klustre enn GapObs på tilfeldige datasett, samtidig som det ikke velges for mange klustre. Metoden RecGapObs finner dermed et bedre estimat for det optimale antall klustre.

Tabell 8.1: Resultater fra GapObs og RecGapObs på de samme simulerte datasettene. Tabellene viser at RecGapObs gjennomsnittlig estimerer et høyere antall klustre for datasettene enn GapObs.

| $k$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| GapObs : | 0 | 0 | 90 | 0 | 1 | 9 | 0 | 0 | 0 | 0 |
| RecGapObs : | 0 | 0 | 2 | 0 | 85 | 13 | 0 | 0 | 0 | 0 |

(a) Resultat fra GapObs og RecGapObs på det simulerte datasettet. Det er tilsammen gjort 100 kjøringer av hver metode, der det samme simulerte datasettet er brukt for alle kjøringer. Se beskrivelse i avsnitt 8.1.

| $k$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| GapObs : | 0 | 7 | 20 | 40 | 28 | 5 | 0 | 0 | 0 | 0 |
| RecGapObs : | 0 | 0 | 3 | 34 | 40 | 23 | 0 | 0 | 0 | 0 |

(b) Resultater fra GapObs og RecGapObs på 100 simulerte datasett. Datasettene er basert på 100 ulike individsimuleringer fra samme klustersentre, se beskrivelse i avsnitt 8.1.

| $k$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| GapObs : | 0 | 6 | 24 | 47 | 20 | 3 | 0 | 0 | 0 | 0 |
| RecGapObs : | 0 | 0 | 10 | 32 | 48 | 10 | 0 | 0 | 0 | 0 |

(c) Resultater fra GapObs og RecGapObs på 100 simulerte datasett. Datasettene er basert på 100 ulike klustersentre, se beskrivelse i avsnitt 8.1.

# Kapittel 9

# Diskusjon

Et overordnet mål for denne oppgaven har vært å teste metoder og å utarbeide nye metoder for å kunne gjenskape resultatene fra Sørlie *et al.* (2003) ved automatikk. Resultatet fra dette studiet var identifikasjonen av fem klustre av individer fra intrinsic Sørlie-datasettet.

For å oppnå dette resultatet ved automatikk, måtte vi ta utgangspunkt i hele intrinsic Sørlie-datasettet og ikke bruke noen subjektive inngrep for å gruppere individer. Det første forsøket på å oppnå dette var å teste gapobservatoren på datasettet. Resultatet fra dette viste at datasettet inneholdt to tydelige klustre og dette resultatet kan ikke anses som godt nok selv om brystkreft lite trolig kan kan deles i et begrenset antall helt veldefinerte subtyper.

Videre analyse av gapobservatoren, ved hjelp av simulerte datasett rundt sentroider fra klustrene i Sørlie *et al.* (2003), viste at metoden klarte å skille klustre relativt godt så lenge standardavviket innenfor hvert kluster var begrenset (under 2.0). Det ble gjort ulike simuleringer med samme varians for hvert kluster, ulikt tilpasset varians for hvert kluster og med uteliggere uniformt fordelt blant dataverdiene. For å sammenlikne simuleringene og intrinsic Sørlie-datasettet, forventet vi at intrinsic Sørlie-datasettet oppdelt i klustre rundt de samme fem sentroidene, hadde et gjennomsnittlige standardavvik innenfor hvert kluster på over 2.0. Dette ble forventet med bakgrunn i tabell 6.2 på side 49, som gir standardavvikene for hvert kluster fra de selekterte individene i intrinsic Sørlie-datatsettet som i Sørlie *et al.* (2003) er tilordnet et kluster. Et estimat på 2 klustre for dette datasettet var derfor ikke overraskende i forhold til spredningen i datasettet. Samtidig kan vi ved hierarkisk klustring se strukturer som tilsier at vi også ved automatisk klustring burde kunne finne et høyere antall klustre.

Gapobservatoren ga altså ikke godt nok resultat, så vi søkte alternative tilpas-

ninger av denne metoden. Det ble foreslått flere modifiserte versjoner av metoden, og en vellykket versjon må sies å være den rekursive tilpasningen av gapobservatoren, RecGapObs. Ved å ta utgangspunkt i resultatet fra gapobservatoren, gjentar denne metoden bruken av gapkriteriet (5.2) på resultatklustrene fra gapobservatoren. Dette blir gjort rekursivt som beskrevet i algoritme 1 på side 59. RecGapObs ga da et estimert antall klustre i intrinsic Sørlie-datasettet lik 5. Dette var et godt resultat fordi det ble oppnådd ved automatikk og resulterte i det samme antallet klustre som var biologisk bestemt i Sørlie *et al.* (2003).

Selv om vi var fornøyd med dette resultatet ville vi teste stabiliteten og robustheten til metoden. Vi gjorde derfor tre ulike simuleringer av datasett som vi testet på både GapObs og på RecGapObs. Den generelle trenden i disse resultatene viste at den rekursive metoden ga et bedre estimat enn GapObs.

## 9.1 Mulig videre arbeid

Videre arbeid vil være å teste stabiliteten og robustheten til RecGapObs ytterlige. En interessant test vil kunne være å finne grensen for hvor mye spredning fra klustersentroidene metoden tåler før den bryter sammen. Dette forsøket vil kunne sammenliknes med de samme testene som ble gjort på GapObs.

Andre muligheter for videre arbeid kan være å implementere de andre alternative tilpasningene nevnt i avsnitt 5.3 på side 38 og videre sammenlikne disse med metodene implementert i denne oppgaven.

En naturlig fortsettelse, etter å ha estimert et antall klustre, vil være å se på kvaliteten av klustringen. Dette kan gjøres i en prosess kalt klustervalidering, som bruker ulike kluster-kvalitetsmål. Et eksempel på dette er IGP (In-group proportions) (Kapp & Tibshirani, 2006). IGP angir grovt sett andelen observasjoner i et kluster som har nærmeste naboer i samme kluster. Andre eksempler er Homogeneity Score, Separation Score, Silhouette Width og Weighted Average Discrepant Pairs (se for eksempel Kapp & Tibshirani, 2006).

En god implementasjon av en estimator for å finne antall klustre i datasett av den typen brukt i oppgaven og en metode for å validere dette resultatet, vil til sammen være et godt verktøy for blant annet medisinsk forskning på store datasett. Dette vil kunne gjøre det mulig å enklere gruppere pasientdata og vurdere pasientenes gruppering med hensyn til diagnose, prognose eller behandling.

# Kapittel 10

# Kildekode

Kildekoden i følgende avsnitt er skrevet i Matlab.

## 10.1 GapObs

Denne metoden antar at data er gitt med observasjoner som rader og variable som kolonner. I oppgaven har jeg vært interessert i å klustre individer, så jeg har lagt individene som rader og genene som kolonner i datasettene. Resultatet av metoden er et estimert antall klustre for datasettet.

Data må være radsentrert og normalisert på forhånd. Dette er gjort med hjelpefunksjonen *sentrer_rad*, se avsnitt 10.3.1 på side 75.

Metoden er hentet fra Tibshirani *et al.* (2001) og er nærmere beskrevet i avsnitt 5.1 på side 33.

Et eksempel på bruk:

```
>> data = sentrer_rad(data);
>> res = gapObs(data, 10, 100)
```

### 10.1.1 gapObs.m

Funksjonen sentrerer først kolonnene i data $X$, noe som er nødvendig fordi vi skal bruke PCA. Deretter kalles funksjonen *getWk*. (Se avsnitt 10.1.2 på side 73.) Denne returnerer innen-gruppe spredningen i $X$. Vi finner så prinsipalkomponentene og genererer deretter referansedata i prinsipalretningene. Referansedata blir så transformert tilbake og vi kan beregne innen-gruppe spredningen. Deretter beregnes gapverdiene etter gapkriteriet gitt i uttrykket(5.2).

Kildekode:

```
function res = gapObs(X, K, B)

n = size(X, 1);
p = size(X, 2);

% Steg 0
Xc = (eye(n) - ones(n,n)./n)*X;

% Steg 1
Wk0 = getWk(X, K);

% Step 2
[U,D,V] = svd(Xc, 0);
Yc = Xc * V;

% Steg 3
Yct = zeros(n, p);
WkB = zeros(K, B);
for b=1:B

    for j = 1:p
      aj = min(Yc(:,j));
      bj = max(Yc(:,j));
      Yct(:,j) = unifrnd(aj, bj, n, 1);
    end

    Xb = Yct * V' + ones(n,n)./n * X;
    WkB(:,b) = getWk(Xb, K);
end

% Steg 4
nullWk = mean(log(WkB), 2);
obsWk = log(Wk0);
Sgap = nullWk - obsWk;

Sdgap = std(log(WkB), 0, 2);
dif1 = Sgap-Sdgap;
dif2 = Sgap+Sdgap;

Sdgap = Sdgap(2:K);
Sgap1 = Sgap(1:K-1);
Sgap2 = Sgap(2:K);

% Steg 5
finder = find(Sgap1 >= ( Sgap2 - Sdgap/2 ))
if numel(finder) > 0
    res = finder(1);
else
    error = 'Det finnes intet kluster som tilfredsstiller gapobservatorkriteriet'
end
```

```
kp = 1:K;
textout = sprintf('Gap-funksjonen. Data: %d x %d. khat = %d.', p, n, res);
plot(kp, Sgap, '-', kp, dif1, '+', kp, dif2, '+'); title(textout);
```

### 10.1.2 getWk.m

Funksjonen beregner innen-gruppe spredningen $W_k$, gitt i uttrykket (5.1) på side 34. Dette er kort oppsummert summen av indreavstandene mellom alle punkter i alle klustre for $k = 1, \ldots, K$ antall tillatte klustre.

Radene i datasettet blir klustret. Klustringsmetoden er agglomerativ hierarkisk klustring med average linkage og det brukes euklidsk avstandsmål.

Kildekode:

```
function res = getWk(X, K)

Wk = zeros(K, 1);

Y = pdist(X, 'euclidean');
Z = linkage(Y, 'average');
d = squareform(Y).^2;

for k=1:K
    Dr = zeros(k, 1);
    nr = zeros(k, 1);
    C = cluster(Z, 'maxclust', k);

    for r=1:k
      Dr(r) = sum(sum(d(C==r, C==r)));
      nr(r) = sum(C==r);
    end

    Wk(k) = sum(Dr./(2*nr)) ;
end

res = Wk;
return
```

## 10.2 RecGapObs

Denne metoden er avhengig av funksjonene *gapObs*, *getWk* og *sentrer_rad*. Metoden er rekursiv og det antas at radene i data skal klustres. Parameteren *names* er valgfri. Det vil si at den kan gis som en tom matrise ([]), eller som en vektor av samme lengde som antall individer, med individ-id'er.

Metoden kan i tillegg til å returnere estimert antall klustre i datasettet, returnere individ-id'ene sortert etter klustertilhørighet og korrelasjonen mellom hvert individ og sin klustersentroide. Dette returneres kun hvis parameteren *names* ikke er tom.

### 10.2.1 recGapObs.m

Funksjonen sentrerer og standardiserer først radene i dataene, for deretter å kjøre metoden GapObs. Etter dette kjøres samme klustringsmetode som i GapObs , for å gjenfinne klustrene derfra. Det sjekkes så om resultatet fra GapObs er større eller lik 1. Hvis det er større enn 1 gjøres et rekursivt kall på RecGapObs , hvis det er lik 1 returneres verdien 1. I tillegg returneres individ-id'er og korrelasjoner for individene i klusteret, hvis parameteren *names* er gitt.

Et eksempel på bruk:

> > [res, res_names, res_cor] = recGapObs(data, 10, 100, rowheaders)

Kildekode:

```
function [res, res_names, res_cor] = recGapObs(X, K, B, names)

X = sentrer_rad(X);
if(size(X,1) ~= 1)
    result = gapObs(X, K, B);
    Y = pdist(X, 'euclidean');
    Z = linkage(Y, 'average');
    C = cluster(Z, 'maxclust', result);
else
    result = 1;
end

res = 0;
res_names = 0;
res_cor = 0;

if(isempty(names) == 0)
    if(result > 1)
        res_names = cell(50, 1);
        res_cor = zeros(50, 1);
        for i=1:result
           [rec_res, rec_names, rec_cor] =  recGapObs(X(C==i,:), K, B, names(C==i));
           res = res + rec_res;
           if(i==1)
               res_names = rec_names;
               res_cor = rec_cor;
           else
```

```
                res_names = [res_names, rec_names];
                res_cor = [res_cor, rec_cor];
            end
        end
    elseif(result == 1)
        res = 1
        res_names = cell(50, 1);
        res_names(1:size(names,1)) = names
        res_cor = zeros(50, 1);
        cor = corr(X', mean(X,1)');
        res_cor(1:size(cor,1)) = cor;
    end
else
      if(result > 1)
        for i=1:result
           rec_res =  recGapObs(X(C==i,:), K, B, []);
           res = res + rec_res;
        end
    elseif(result == 1)
        res = 1;
    end
end
```

## 10.3 Hjelpefunksjon

### 10.3.1 sentrer_rad.m

Denne funksjonen sentrerer først radene i det gitte datasettet for så å normalisere radene.

Kildekode:

```
function res = sentrer_rad(X)

p = size(X, 2);

X = X*(eye(p) - ones(p,p)./p);

rad_norm = [];
for n=1:size(X,1)
  rad_norm = [rad_norm; norm(X(n,:))];
end

res = X./repmat(rad_norm, 1, size(X,2));
```

# Bibliografi

## Trykte referanser


**Brown T A.** *Genomes* 3rd edition. University of Manchester, United Kingdom. Garland Science, 2007.

**Campbell N A, Reece J B.** *Biology* 6th edtition. San Francisco, California. Benjamin Cummings, 2002.

**Causton H C, Quackenbush J, Brazma A.** *Microarray/Gene Expression Data Analysis: A Beginner's Guide.* Blackwell 2003.

**Draghici S.** *Data analysis tools for microarraydata.* Chapman & Hall/CRC, London 2003.

**Forus A, Sørlie T, Børresen-Dale A-L, Myklebost O.** *Mikromatriser i kreftforskning - nå trenger vi ikke lete bare under gatelyktene!* Tidsskrift for den norske Lægeforening nr 21, 2001; 121: 2498.

**Hastie T, Tibshirani R, Eisen M B, Alizadeh A, Levy R, Staudt L, Chan W C, Botstein D, Brown P.** *'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns.* Genome Biology 2000, vol 1(2):Research0003.

**Hastie T, Tibshirani R, Friedman J.** *The Elements of Statistical Learning; Data Mining, Inference and Prediction.* Springer 2001.

**Hirano T.** *Cell biology: Holding sisters for repair.* Nature 3. feb 2005, vol 433, side 467-468

**Huang H-X, Liang Z-A, Pardalos P M.** *Some Properties for the Euclidean Distance Matrix and Positive Semidefinite Matrix Completion Problems.* Journal of Global Optimization, Januar 2003, vol. 25, no 1, side 3-21.

**Jenssen T K.** *Bioinformatics methods for analysis of gene expression data.* Doktoravhandling 2002:8 NTNU Trondheim.

**Kapp A, Tibshirani T.** *Are cluster in one dataset present in another dataset?* Biostatistics, 12. april 2006.

**Kerr M K, Churchill G A.** *Statistical design and the analysis of gene expression microarray data.*
Genetical Research, april 2001, 77(2): 123-128.

**Kohonen T.** *Self-Organizing Maps* Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York, 1995, 1997, 2001.

**Lander E S, Linton L M, Birren B, Nusbaum C, Zody M C, Baldwin J, Devon K, Dewar K, Doyle M, FitzHugh W.** *et al.* International Human Genome Sequencing Consortium. *Initial Sequencing and analysis if the human genome.*
Nature 15. februar 2001, vol 409, side 860-921.

**Lay D C.** *Linear Algebra and its applications, 3rd edition.* Pearson Education 2005.

**Lee M-L T, Kuo F C, Whitmore G A, Sklar J.** *Importance of replication in microarray gene expression studies: Statistical methods and evidence from repetitive cDNA hybridizations.* PNAS, 29. August 2000, vol. 97, no. 18, side 9834-9839.

**Lockhart D J, Winzeler E A.** *Genomics, gene expression and DNA arrays.* Nature 2000, Vol 405, 827 - 836

**Lingjærde O C.** Institutt for Informatikk, Universitet i Oslo. *Lysark til forelesning i INF2300, 11.04.2005.*
http://www.ifi.uio.no/forskning/grupper/bioinf/Teaching/INF2300
/Forelesninger/inf2300-lec9-4prpage.pdf

**Moore D S, McCabe G P.** *Introduction to the practice of statistics, 4th edition.* W.H. Freeman and Company 2003.

**Sandvik AK, Støren O, Nørsett K, Lægreid A, Børresen-Dale AL, Myklebost O.** *Måling av genaktivitet med DNA-mikromatriser.* Tidsskrift for den norske Lægeforening nr 10, 2001, 121: 1225.

**Smolkin M, Ghosh D.** *Cluster stability scores for microarray data in cancer studies.* BMC Bioinformatics 2003, 4:36.
http://www.biomedcentral.com/1471-2105/4/36

**Speed T.** *Statistical Analysis of Gene Expression Microarray Data.* Chapman & Hall/CRC, 2003, pp. 190-200.

**Sørlie T, Perou C M, Tibshirani R, Aas T, Geisler S, Johnsen H, Hastie T, Eisen M B, Rijn M van de, Jeffrey S S *et al.*** *Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications* PNAS, 11. september 2001, vol. 98, no. 19, 10869-10874.

**Sørlie T, Tibshirani R, Parker J, Hastie T, Marron J. S., Nobel A, Deng S, Johnsen H, Pesich R, Geisler S, *et al.* .** *Repeated Observation of Breast Tumor Subtypes In Independent Gene Expression Data Sets.* PNAS, 8.juli 2003, vol. 100, no. 14, 8418-8423
http://genome-www.stanford.edu/breast_cancer/robustness/

http://smd.stanford.edu/cgi-bin/publication/viewPublication.pl?pub_no=248
Aksess 19.april 2007

**Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, Dmitrovsky E, Lander E S, Golu T R.** *Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation.* Proc Natl Acad Sci U S A. 1999 Mar 16;96(6):2907-12.
http://www.pnas.org/cgi/content/full/96/6/2907

**Tempelman R J.** *Assessing statistical precision, power, and robustness of alternative experimental designs for two color microarray platforms based on mixed effects models.* Veterinary Immunology and Immunopathology 2005, vol 105, side 175 - 186.

**Tibshirani R, Walther G, Hastie T.** *Estimating the number of clusters in a data set via the gap statistic.* Royal Statistical Society, 2001. 63, Part 2, p 411-423.

## Nettsider

**Breastcancer.org**
http://www.breastcancer.org/symptoms/diagnosis/staging.jsp
Aksess 5.9.2007

**Human Genome Program** *Genomics and Its Impact on Science and Society; The Human Genome Project and Beyond.* A publication of the U.S. Department of Energy, Human Genome Program. Mars 2003.
http://www.ornl.gov/sci/techresources/Human_Genome/publicat/primer2001/1.shtml
Aksess 31.8.2007

**Human Genome Project**
http://www.ornl.gov/sci/techresources/Human_Genome/project/info.shtml
Sist oppdatert 18. nov 2005.

**HybridHclust**
Eisencluster algorithm.
http://cran.r-project.org/doc/packages/hybridHclust.pdf
Aksess 25.10.2007

**Jornsten R** Implementasjon av gapobservatoren i R (*gappcalg.R*).
http://www.stat.rutgers.edu/ rebecka/RCode/
Aksess 19.04.2007

**Matlab** Matlab versjon 7.4. R2007a.
http://www.mathworks.com/
Aksess 10.9.2007

**MedicineNet.com** Medical Dictionary.
http://www.medterms.com/script/main/hp.asp
Aksess 5.9.2007

**NCBI** National Center for Biotechnology Information - A Science Primer.
http://www.ncbi.nlm.nih.gov/About/primer/microarrays.html
Sist oppdatert 24.03.2004.

**NHGRI** National Human Genome Research Institute
http://www.genome.gov/glossary.cfm?
Aksess 25.10.2007

**NIH** National Institute of Health, The Genomics & Bioinformatics Group.
http://discover.nci.nih.gov/microarrayAnalysis/Experimental.Design.jsp
25.10.2007

**NuGO** The European Nutrigenomics Organisation.
http://www.nugo.org/everyone
Aksess 25.10.2007

**R** Statistikk-programmiljøet R.
http://www.r-project.org/
Aksess 5.9.2007

**Resampling Stats** http://www.resample.com/
Aksess 14.3.2006.

**Skjermdumper** Skjermdumper av valg for nedlasting av datasett fra Stanford Microarray Database.
http://heim.ifi.uio.no/espes/master/dataset/dataretrieval0509.doc
Aksess 24.09.2007

**Stanford Medicine Magazine**
http://mednews.stanford.edu/stanmed/2005fall/microarray.html
Aksess 25.10.2007

**SMD** Stanford Microarray Database
http://genome-www5.stanford.edu/
Aksess 24.09.2007

**The National Academies Press**
http://www.nap.edu/
Aksess 25.10.2007

**NLM** U.S. National Library of Medicine
http://ghr.nlm.nih.gov/
Aksess 25.10.2007